Navigator User Guide

Informatica® PowerExchange®
(Version 8.6.1 HotFix 11)

Informatica PowerExchange Navigator User Guide

Version 8.6.1
May 2010

Part Number: PWX-NUG-861-0003

Table of Contents

Preface

This guide describes how to use the PowerExchange Navigator to build and maintain your PowerExchange installation.

This manual applies to the following PowerExchange products:

- PowerExchange for Adabas®
- PowerExchange for CA Datacom®
- PowerExchange for CA IDMS™
- PowerExchange for DB2® for i5/OS®
- PowerExchange for DB2 for Linux®, UNIX®, and Windows®
- PowerExchange for DB2 for z/OS®
- PowerExchange for IMS™
- PowerExchange for Oracle®
- PowerExchange for SQL Server®
- PowerExchange for VSAM

The PowerExchange Navigator runs on Windows. It consists of the following main components:

- **Data capture**, if installed. Definitions for capture registrations and extraction maps for PowerExchange Change Data Capture (CDC).
- **Data maps.** The relational to nonrelational definitions that are used to access data sources like IMS databases, VSAM files and flat files.
- **Personal metadata**. Parameters to browse metadata from remote databases.

Informatica Resources

Informatica Customer Portal

As an Informatica customer, you can access the Informatica Customer Portal site at http://my.informatica.com. The site contains product information, user group information, newsletters, access to the Informatica customer support case management system (ATLAS), the Informatica How-To Library, the Informatica Knowledge Base, the Informatica Multimedia Knowledge Base, Informatica Documentation Center, and access to the Informatica user community.

Informatica Documentation

The Informatica Documentation team takes every effort to create accurate, usable documentation. If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com. We will use your feedback to improve our documentation. Let us know if we can contact you regarding your comments.

The Documentation team updates documentation as needed. To get the latest documentation for your product, navigate to the Informatica Documentation Center from http://my.informatica.com.

Informatica Web Site

You can access the Informatica corporate web site at http://www.informatica.com. The site contains information about Informatica, its background, upcoming events, and sales offices. You will also find product and partner information. The services area of the site includes important information about technical support, training and education, and implementation services.

Informatica How-To Library

As an Informatica customer, you can access the Informatica How-To Library at http://my.informatica.com. The How-To Library is a collection of resources to help you learn more about Informatica products and features. It includes articles and interactive demonstrations that provide solutions to common problems, compare features and behaviors, and guide you through performing specific real-world tasks.

Informatica Knowledge Base

As an Informatica customer, you can access the Informatica Knowledge Base at http://my.informatica.com. Use the Knowledge Base to search for documented solutions to known technical issues about Informatica products. You can also find answers to frequently asked questions, technical white papers, and technical tips. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.

Informatica Multimedia Knowledge Base

As an Informatica customer, you can access the Informatica Multimedia Knowledge Base at http://my.informatica.com. The Multimedia Knowledge Base is a collection of instructional multimedia files that help you learn about common concepts and guide you through performing specific tasks. If you have questions, comments, or ideas about the Multimedia Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.

Informatica Global Customer Support

There are many ways to access Informatica Global Customer Support. You can contact a Customer Support Center through telephone, email, or the WebSupport Service.

Use the following email addresses to contact Informatica Global Customer Support:

- support@informatica.com for technical inquiries
- support_admin@informatica.com for general customer service requests

WebSupport requires a user name and password. You can request a user name and password at http://my.informatica.com.

Use the following telephone numbers to contact Informatica Global Customer Support:

North America / South America	Europe / Middle East / Africa	Asia / Australia
Informatica Corporation Headquarters 100 Cardinal Way Redwood City, California 94063 United States **Toll Free** +1 877 463 2435 **Standard Rate** Brazil: +55 11 3523 7761 Mexico: +52 55 1168 9763 United States: +1 650 385 5800	**Informatica Software Ltd**. 6 Waltham Park Waltham Road, White Waltham Maidenhead, Berkshire SL6 3TN United Kingdom **Toll Free** 00 800 4632 4357 **Standard Rate** Belgium: +32 15 281 702 France: +33 1 41 38 92 26 Germany: +49 1805 702 702 Netherlands: +31 306 022 797 Spain and Portugal: +34 93 480 3760 United Kingdom: +44 1628 511 445	**Informatica Business Solutions Pvt. Ltd.** Diamond District Tower B, 3rd Floor 150 Airport Road Bangalore 560 008 India **Toll Free** Australia: 1 800 151 830 Singapore: 001 800 4632 4357 **Standard Rate** India: +91 80 4112 5738

CHAPTER 1

Using the PowerExchange Navigator

This chapter includes the following topics:

PowerExchange Navigator Overview

Many of the world's largest enterprises have vast amounts of information locked away in mainframe and open systems. Often, a requirement exists to move this data into another format, typically for use in a data warehouse, data mart, or a decision support system. Typically, moving this data requires the following steps:

1. Extract data from the operational system.
2. Translate data on the source or target system.
3. Move data to the target system.
4. When the data arrives, load it into the application, such as a relational database.

PowerExchange, in conjunction with PowerCenter, eliminates these steps and completes the process in one step. PowerExchange not only extracts source data, it also moves the data at a high speed and loads it into target databases. You can initiate this processing from any platform. Because PowerExchange is not restricted to vendor-specific solutions like ODBC, no prerequisite software is required to access any database.

Use the PowerExchange Navigator to build and maintain PowerExchange source and target definitions. It consists of the following main components:

- **Data Capture.** Use the Data Capture folder to define and manage capture registrations and extraction maps for PowerExchange Change Data Capture (CDC).
- **Data Maps.** Use the Data Maps folder to define and manage nonrelational data maps to access data sources like IMS databases, VSAM files, and flat files. You can also define DB2 data maps.
- **Personal Metadata**. Use the Personal Metadata folder to browse remote database metadata and data.

Preparing to Use the PowerExchange Navigator

The PowerExchange Navigator runs on Windows. The PowerExchange Listener is installed as a Windows Service and communicates status information through the taskbar notification area.

Installing PowerExchange on Windows

Before using the PowerExchange Navigator, complete the following prerequisites:

- Install the Windows version of PowerExchange. This action installs PowerExchange Navigator, a necessary component for sourcing data, irrespective of where the source or target databases are located.
- Restart the computer.

Additionally, you must have a basic understanding of Listeners and issues like COBOL copybooks and their usage.

Starting the PowerExchange Listener on Windows

Use options in the Informatica PowerExchange Program Group to start and stop the PowerExchange Listener service. When active, the PowerExchange Listener puts an icon in the System Tray. If this fails to appear, a problem exists and the PowerExchange detail.log file must be inspected. This file is updated in the current working directory when PowerExchange is running. The service is considered to start in the system directory so it is likely to be found in the C:\winnt\system32 directory or the equivalent name on your system.

- To stop the PowerExchange Listener service, select Stop PowerExchange Listener from the Informatica PowerExchange Program Group. Services can also be started and stopped from the Services dialog box.
- To start the PowerExchange Listener manually from a command prompt dialog box, issue the command:

  ```
  dtllst node1
  ```

  As it starts it displays messages to the console showing which port is being polled and after starting completely it issues the PWX-00607 message. It is now ready for communication. Enter c, to close it from the keyboard.

Note: You do not need to run the PowerExchange Listener on Windows systems to define data maps and capture registrations. You only need to run the PowerExchange Listener on the Windows system if you are extracting data from this machine using a PowerExchange Listener on another system.

Testing the Windows Listener

Go to the Program Group typically called PowerExchange and a BAT file icon for DTLPING is displayed. This runs a PowerExchange Client ping, which is a test that the connection to the PowerExchange Listener is alive. The loc=node1 is set in the shipped PowerExchange configuration file, dbmover.cfg, to use the TCP/IP Loopback IP address of 127.0.0.1. Contact should be made with the PowerExchange Listener and the request should complete with a zero return code.

If it does not, research the reason for failure. The error messages output by PowerExchange contain a lot of information. The most likely causes of failure are the IP address and port number on the NODE and PowerExchange Listener for node1 in the PowerExchange configuration file, dbmover.cfg, being incorrect. Other program loading issues are likely to be set-up errors. The steps taken during the install should be revisited to ensure that they all completed successfully with zero return codes.

Testing a Remote Listener

If you are going to use a remote PowerExchange Listener, at some stage you are going to need to start it. After you start it, check whether it is started, and whether connectivity is established by entering the following command:

```
dtlrexe loc=xxxx prog=ping
```

The *xxxx* variable is the NODE name in the PowerExchange configuration file, dbmover.cfg, pointing to the remote platform.

Configuring ODBC Data Sources on Windows

On Windows, if you want to use ODBC data sources to access local or remote sources, use the following procedure.

Tip: To integrate PowerCenter with PowerExchange, Informatica recommends that you use the PowerExchange Client for PowerCenter (PWXPC) instead of PowerExchange ODBC. PWXPC offers additional functionality. For more information, see *PowerExchange Interfaces for PowerCenter*

To configure an ODBC data source:

1. Install the PowerExchange ODBC drivers.

   For more information, see the *PowerExchange Installation Guide*.

2. Create an ODBC data source on Windows.

   For more information, see the *PowerExchange Reference Manual*.

Preparing to Access Sources and Targets

To use PowerExchange to source data, you need to know specific information about the databases, tables, and files. The purpose of the following roadmap is to take someone with PowerExchange installed on one or more platforms and show them how to tackle various problems related to accessing supported data formats on the local platform or on a remote one.

Sourcing Relational Data

- Obtain the valid user ID and password for the source database.
- Obtain any database specific information, such as:
  - Oracle SID
  - DB2 for z/OS subsystem ID or group name
  - DB2 for Linux, UNIX, and Windows database name and instance name
  - DB2 for i5/OS database name
  - Sybase and Microsoft SQL Server server and database names
- Create any database-specific connectivity, such as Oracle TNS Names.
- Ping the IP address of the database source machine to check it is visible.
- Add a NODE line pointing to the database machine in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a Personal Metadata profile and view some real data, having checked that the metadata is returned successfully.
- Use PowerCenter or an OEM tool to execute the request to move data.

Sourcing MVS VSAM Data Sets and Flat Files from any Platform

- If the file is not a comma or delimited text file, find the file layout, such as a COBOL or PL/1 copybook.
- Find the exact file name on the remote platform.
- Ping the IP address of the source machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map.
- Import a COBOL or PL/1 copybook to define the file layout or manually create it.
- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

Sourcing Data from IMS

- Ping the IP address of the source machine to check it is visible.

If sourcing data using DL/I Batch or BMP:

- Update the MVS PowerExchange configuration file to add a second PowerExchange Listener entry and a NETPORT line with the correct PSB name.

  Note: PowerExchange does not support PSBs coded with LANG=PLI.

  Stop and restart the MVS PowerExchange Listener.
- Update the IMSJCL member in the MVS PowerExchange RUNLIB PDS. The name of this file is also defined in the NETPORT line updated previously.
- Add a NODE line pointing to the MVS NETPORT PowerExchange Listener entry in the local Windows PowerExchange configuration file.

If sourcing data using IMS ODBA:

- Ensure RESLIB data set is allocated to the PowerExchange Listener, such as STEPLIB or MVS LNKLST concatenation.
- Use the PSB which has PCBNAMES defined.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- To create a data map, gather the IMS SSID, PSB, and PCB names. Use the PowerExchange Navigator to create a data map.
- If a DBD is available, import it to define the segments and the hierarchical sequence.
- If no DBD was to be imported, or there are COBOL copybooks defining the segment layouts, import them, possibly deleting any DBD generated ones.
- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

Sourcing Data from Adabas

- Ping the IP address of the source machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- Use the following command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  This command tests that the remote PowerExchange Listener is started, where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map. Select Adabas as the access method in the map wizard and specify the Adabas database ID and file number.
- Decide which Adabas data definition to import. The PowerExchange Navigator can import from a variety of sources:
  - DDM
  - DDM OPEN SYS (Open System)
  - PREDICT
  - FDT
  - ADACMP
  - TEXT.

  It is also possible to import COBOL and PL/1 copybooks and manually associate the PowerExchange field definitions with Adabas field definitions.
- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

Sourcing Data from IDMS

- Ping the IP address of the source machine to check it is visible.
- Ensure DBMOVER configuration member on MVS points to the correct library containing the IDMS metadata JCL IDMSMJCL (LOADJOBFILE parameter).
- Customize the IDMSMJCL member for your installation.
- Customize the MVS PowerExchange Listener procedure/job for your installation. Ensure it is updated to allow IDMS remote or local access.
- Start the MVS PowerExchange Listener.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map. Select IDMS as the access method in the map wizard and specify the Sub Schema and Dictionary names.
- Perform a remote import. It is also possible to manually create PowerExchange record definitions or import COBOL and PL/1 copybooks. They can then be manually associated with IDMS record, area and set definitions.
- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

Sourcing Data from Datacom

- Ping the IP address of the source machine to check it is visible.
- Customize the MVS PowerExchange Listener procedure/job for your installation.
- Start the MVS PowerExchange Listener.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map. Select Datacom as the access method in the map wizard and specify the Database name and ID.
- Perform a remote import. It is also possible to manually create PowerExchange record definitions or import COBOL and PL/1 copybooks.
- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

Targeting MVS VSAM Data Sets or Flat Files on any Platform

- Find the layout of the file to be created, such as a COBOL or PL/1 copybook.
- Decide on the exact file name on the remote platform.
- If it is an MVS system, preallocate the file.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map.
- Import a COBOL copybook to define the file layout or manually create it.
- Do a row test even though it is likely that no data is present. This action is required to send the data map to the remote platform.
- Use PowerCenter or an OEM tool to execute the request to move data.

Targeting Adabas

- Determine the Adabas database ID and file number.
- Ping the IP address of the source machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map. Select Adabas as the access method in the map wizard and specify the Adabas database ID and file number.

- Decide which Adabas data definition to import. The PowerExchange Navigator can import from a variety of sources:
  - DDM
  - DDM OPEN SYS (Open System)
  - PREDICT
  - FDT
  - ADACMP
  - TEXT

  It is also possible to import COBOL and PL/1 copybooks and manually associate the PowerExchange field definitions with Adabas field definitions.
- Do a row test to return some valid data and run this process as many times as necessary, making changes to the data map until the data returned is as required.
- Use PowerCenter or an OEM tool to execute the request to move data.

Targeting IMS

- Find the layout of the file to be created.
- Decide on the exact file name on the remote platform.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map.
- Manually define the file layout or import a copybook.
- Do a row test even though it is likely that no data is present. This action is required to send the data map to the remote platform.
- Use PowerCenter or an OEM tool to execute the request to move data.

Targeting C-ISAM Data

- Find the layout of the file to be created.
- Decide on the exact file name on the remote platform.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  Where *xxxx* is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a data map.
- Manually define the file layout or import a copybook.
- Do a row test even though it is likely that no data is present. This action is required to send the data map to the remote platform.
- Use PowerCenter or an OEM tool to execute the request to move data.

Targeting Relational Data

- Obtain the valid user ID and password for target database.
- Obtain any database specific information, such as Oracle SID, DB2 SSID, and Sybase Server.
- Create any database specific connectivity, such as Oracle TNS Names.
- Ping the IP address of the target machine to check it is visible.
- Add a NODE line pointing to the remote platform in the PowerExchange configuration file.
- To test that the remote PowerExchange Listener is started, use the command:

  ```
  dtlrexe loc=xxxx prog=ping uid=userid pwd=pwd
  ```

  The *xxxx* variable is the NODE name from the PowerExchange configuration file.
- Use the PowerExchange Navigator to create a Personal Metadata profile and view some real data, having checked that the metadata is returned successfully.
- Use PowerCenter or an OEM tool to execute the request to move data, setting the table creation parameters as required.

Defining the Resource Configuration

You can add or edit a resource configuration, which identifies the location of your data maps, command sets, and personal metadata. If you have defined multiple resource configurations to separate test and production environments, you can switch to a specific resource configuration.

Data maps and command sets are shareable, while personal metadata is not. Personal metadata is maintained in the local path, while data maps and command sets can be stored in the local path or a shared path.

In the Resource Configuration dialog box, you can perform the following actions:

- To switch to a specific resource configuration, click the Current Configuration tab. See “Current Configuration Tab” on page 8.
- To add, edit, or delete a resource configuration, click the Edit Configuration tab. See “Edit Configuration Tab” on page 9.

Current Configuration Tab

If you have defined multiple resource configurations, you can switch to a configuration by using the Current Configuration tab.

To switch to a PowerExchange resource configuration:

1. In the PowerExchange Navigator, click **Options > Resource Configuration**.
2. On the Resource Configuration dialog box, click the Current Configuration tab.
3. From the **Name** list, select a resource configuration.
4. Click **OK**.

Edit Configuration Tab

You can add, edit, or delete a resource configuration by using the Edit Configuration tab.

- If you are starting PowerExchange Navigator for the first time, the Add Configuration dialog box is displayed automatically. Proceed to step 4 in the following procedure to select the location of your data maps, command sets, and personal metadata.
- If registry entries exist from a previous installation, you are not automatically prompted to add a resource configuration. Instead, use the following procedure to add, edit, or delete a resource configuration.

To add, edit, or delete a PowerExchange resource configuration:

1. In the PowerExchange Navigator, click **Options > Resource Configuration**.
2. On the Resource Configuration dialog box, click the Edit Configuration tab.
3. Perform one of the following actions:
   - To add a resource configuration, click **Add**.
   - To edit a resource configuration, select a configuration from the **Available Configurations** list. Then click **Edit**.
   - To delete a resource configuration, select a configuration from the **Available Configurations** list. Then click **Delete**. You cannot delete the default resource configuration.
4. On the Edit Configuration or Add Configuration dialog box, edit the following fields:

Field	Description
Name	Specifies the name of the resource configuration. If you are adding a resource configuration for the first time, the directory that you choose becomes the default.
Local Path	Specifies location of your data maps and other resources. Click the **Browse** button to locate the path. For a local configuration, specify only the **Local Path**. For a shared configuration, specify both the **Local Path** and the **Shared Path**.
Shared Path	Specifies a shared location of your data maps and other resources. Click the **Browse** button to locate the path. Ensure that you have mapped a network drive before specifying the **Shared Path**. For more information about using a shared path, see “Shared Resource” on page 10. For a local configuration, specify only the **Local Path**. For a shared configuration, specify both the **Local Path** and the **Shared Path**. **Notes**: - Specify only data maps and command sets as shared resources. Do not attempt to share personal metadata because this action can cause data corruption. If you specify personal metadata within a shared configuration, it is stored in the local directory. - Access to a shared resource is read-only if that resource is in use by another user.
Set as current configuration	Sets the new resource configuration as the current configuration. **Note**: This option is displayed for the Add Configuration dialog box only.

5. Click **OK**.

Shared Resource

To share data maps or command sets, you must create a shared resource. By using a shared resource, data is maintained in the following locations:

- Data maps and command sets: In the shared path
- Personal metadata: In the local path

Note: To create a shared resource, you must first map a network drive to a local disk drive because the double backslash (\\) is an invalid string in a PowerExchange path name. See the procedure in "To create a shared resource:" on page 10.

Multiple users can access a shared resource, but the shared resource is protected from concurrent updates. The first user to access a shared resource has full access, but subsequent concurrent users are restricted from performing the following actions to the shared resource:

- Saving changes
- Renaming or deleting
- Exporting
- Testing
- Sending to host

If you try to open a shared resource that is already in use by another user, a warning message is displayed.

To create a shared resource:

1. Right-click **My Network Places**, and click **Map Network Drive.**
2. In the Map Network Drive dialog box, select the drive letter for the connection and the folder that you want to connect to.
3. Click **Finish**.
4. In the PowerExchange Navigator, edit the fields on the Edit Configuration or Add Configuration dialog box. Specifically, enter the following information:
   - In the **Shared Path** field, browse to the network drive that you mapped. When you specify both a shared and local path, ensure that these paths differ. Otherwise, an error message is displayed.
   - For a new resource configuration, select the **Set as current configuration** option.
5. Click **OK**.

   After you define a shared resource, the PowerExchange Navigator main window title bar displays the resource configuration in use and indicates that it is shared.

Selecting Views

On the toolbar, click the arrow next to the **Views** icon to select one of the following views:

- **Large Icons**
- **Small Icons** (default)
- **List**
- **Details**

By selecting a view, the viewing layout and style of the content in the **Resources** pane changes.

CHAPTER 2

Creating Data Maps

This chapter includes the following topics:

Data Map Overview

PowerExchange data maps consist of record and table definitions for a source or target. Data maps define the field layout of the records in the source or target. The table definition in the data map provides a relational view of the data.

PowerExchange uses the table definitions in data maps to construct SQL statements to access source and target data. In the case of nonrelational sources and targets, PowerExchange interprets the SQL statements internally. For relational sources and targets, PowerExchange passes the SQL to the RDBMS for processing.

You can test a data map by using a database row test. A database row test accesses actual source data and displays it in table format. You can use a database row test to ensure that data can be retrieved from the source database. For more information, see “Using Database Row Test” on page 141.

Your PowerExchange installation provides sample data files and data maps that you can use in the examples described in “Creating Data Maps - Examples” on page 19.

Data Map Naming and SQL Generation

PowerExchange provides a naming convention for data maps that enables you to group your data maps by projects, platforms, or another category. For example, you might create ims.nrdb01 or project08.test01.

When you add a data map, you specify the following values:

- A schema name, such as **demo**
- A map name, such as **map1**

The data map name is constructed from the schema name and map name, for example, **demo.map1**.

When you perform a database row test or an extraction on a table, PowerExchange generates a SQL statement with the table name prefixed with the schema name and map name.

Depending on naming convention used by the **DB Type** that you select in the Database Row Test dialog box, PowerExchange generates SQL statements in the following manner:

- For DB types that use a three-tier naming convention, the table name is prefixed with the schema name and map name:

```
select * from demo.map1.table1
```

- For DB types that use a two-tier naming convention, the table name is prefixed with the schema name and map name:

```
select * from demo.map1_table1
```

For example, the NRDB DB type uses a three-tier naming convention, while the NRDB2 DB type uses a two-tier naming convention.

To view the naming convention used for the generated SQL, on the Data Map tab, select the data map. Then click **Options > Preferences**. If the **Use 2-tier names** option is selected, a two-tier name is used. If this option is cleared, a three-tier name is used.

Content of a Data Map

You associate a data map with a single data file, which defines the record types and the data in the data map.

In a data map, you define one or more records and tables:

- Define a single record in a data map if the associated data file contains only one record type. In a single-record data map, you can define multiple tables.
- Define multiple records in a data map only if multiple record types exist in the associated data file. The fewer columns that PowerExchange must retrieve results in faster file processing and data extraction.

For examples of creating single-record and a multiple-record data maps, see “Creating Data Maps - Examples” on page 19.

Defining Arrays

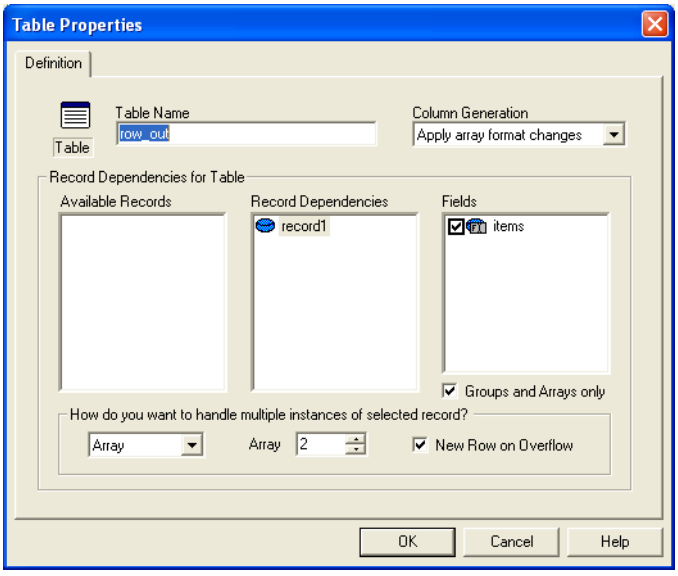
In a data map, a record can contain an array, or in the case of some legacy data sources with OCCURS clauses, multiple arrays.

A simple table can contain a record with one or more arrays. A complex table can contain one or more records, and each record can have one or more arrays.

You can use options in the Table Properties dialog box to define how the elements in an array are:

- Displayed in a database row test
- Mapped to rows and columns in a target table

The following figure shows the Table Properties dialog box:



The following table describes the options in the Table Properties dialog box that you can use to define arrays:

Option	Description
Column Generation	When you first define a table, column names are built automatically by using the record on which the table is based. However, a special naming convention is used for records containing arrays. If you rename columns after you define the table and you do not want column names to be regenerated, select one of the following options: - **Apply array format changes.** Only changes made to array elements are reflected in the table. - **Refresh with missing columns.** Only new columns are added to the table. - **Reset to defaults**. A new set of columns are generated with the default names derived from the field names. Any changes that you have made are discarded. - **Remove Hidden Columns.** For DB2UNLD only. A new set of columns are generated with the default values of **Hide from Table** for each field.
Fields	Select or clear a field to control how elements in an array or group field are displayed: - To display each element in an array or group field in a single row, select the field. - To display each element in an array or group field in a separate row, clear the field. **Note**: To display only fields defined as a group field or as an array in the **Fields** list, select the **Groups and Arrays only** option. To display all fields in the record in the **Fields** list, clear this option. This option controls the fields that are displayed in the **Fields** list and in the Table Properties dialog box, but not the fields that are displayed in the Database Row Test Output window.

Option	Description
How do you want to handle multiple instances of selected records?	**New Row**. A new row is displayed or written to the target for every element in the array.
	Ignore. Second and subsequent elements in the array are not displayed or written to the target.
	Array. The number of elements specified in the **Array** list are displayed or written to the target in a single row of output. PowerExchange populates the output row until it is full, and then either ignores subsequent elements or displays a new row with the overflow elements if the **New Row on Overflow** option is selected.
Multiple Arrays in a Single Input Row	Select this option to generate multiple output rows from a single record that contains multiple arrays, or OCCURS clauses. PowerExchange sets the output fields to NULL when the data in the record is exhausted. For more information about this option, see "Multiple Arrays in Single Input Row" on page 14. This option is enabled only for a table with an imported COPYLIB with multiple OCCURS clauses.

For an example of how to use these options to define how the elements in an array are displayed or mapped, see "Example 2. Single-Record Data Map with an Array" on page 23.

Multiple Arrays in Single Input Row

Many legacy data sources contain multiple arrays, or OCCURS clauses, on a single record. PowerExchange can generate multiple output rows from a single record, and set output fields to NULL when the data in the record is exhausted.

The following table shows some example input:

Input Row	fld1 - OCCURS DEPENDING	fld2 - OCCURS DEPENDING
1	3 values 10,20 30	2 values AA, BB
2	1 value 55	4 values DD1, DD2, DD3, DD4

The following table shows the example output:

Output Row	Fld1	Fld2
1	10	AA
2	20	BB
3	30	NULL
4	55	DD1
5	NULL	DD2
6	NULL	DD3
7	NULL	DD4

You can also use this single input–many output process with other types of fields, such as account numbers, so that every output row is complete.

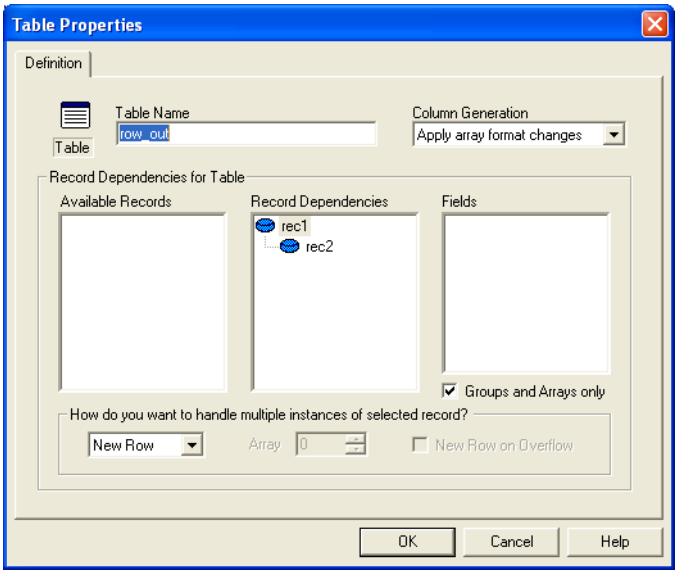
Defining Complex Tables

While a simple table contains one record or segment, a complex table contains more than one record or segment. In a complex table, records or segments can optionally have a defined hierarchical relationship.

In a data map, a complex table mirrors the hierarchy of records or segments in the associated data source, such as an IMS or IDMS database, data set, or a sequential flat file. A complex table combines information into one table that, in a more normalized form, would reside in two or more tables.

You can define complex tables by setting options in the Table Properties dialog box.

When you view table properties for a complex table with a hierarchical relationship, records appear in the **Record Dependencies** list as follows:



For more information, see “Complex Table Options” on page 16.

Supported Data Sources for Complex Tables

The following table lists the data sources, with the associated access methods, for which support complex tables:

Data Source	Access Method	Comments	Reference
CA-IDMS databases	IDMS	For IDMS, a record can have multiple parents.	“Creating IDMS Data Maps” on page 41
IMS databases	IMS ODBA - Uses the ODBA interface	For IMS, the PowerExchange Navigator provides a **Display IMS Hierarchy** option so that you can view the IMS database hierarchy.	“Creating IMS Data Maps” on page 46
	DL/1 BATCH - Uses BMP and DL/1 batch jobs		
Sequential data sets and flat files	SEQ		“Example 3. Multiple-Record Data Map with a Complex Table” on page 26
Tape data sets on MVS	TAPE		
VSAM ESDS data sets	ESDS		“Creating VSAM Data Maps” on page 62
VSAM KSDS data sets	KSDS		

Complex Table Options

You can define complex tables by setting options in the Table Properties dialog box on the Definition and IMS Options tabs.

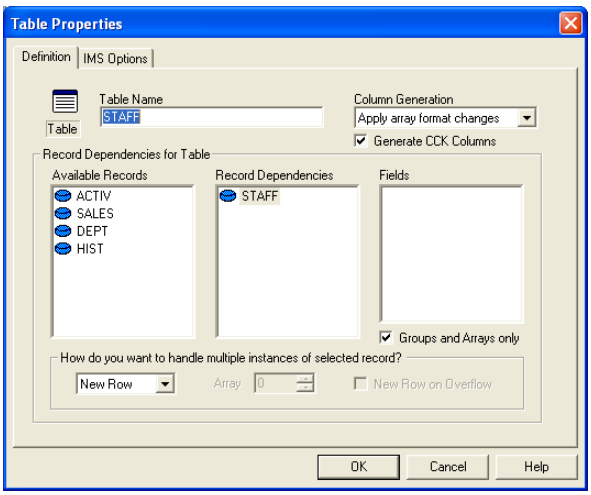
These options control how records in the complex table are:

- Displayed in a database row test
- Mapped to rows and columns in a target table

Definition tab:

Options on the Definition tab in the Table Properties dialog box enable you to define a hierarchy in a complex table, and settings for how records in a complex table are displayed in a database row test or mapped in a target table.

The following figure shows the Definition tab:



The following table describes the options on the Definition tab that you can use to define complex tables:

Option	Description
Available Records	A list of the records in the data map that are not in the complex table. To add a record to the **Record Dependencies** list, in the **Available Records** list, right-click a record and click **Add Record**. To add a child record to a parent record: - In the **Record Dependencies** list, click a record to identify it as the parent record. - in the **Available Records** list, right-click a record and click **Add Record as Child**. This action moves the record to the **Record Dependencies** list as a child record of the parent record. **For IDMS only**: To select the system index to be used for record retrieval, right-click a record and click **Use System Index**. Then, select the index.
Record Dependencies	A list of the records that are in the complex table, with any defined hierarchical dependencies. To remove a record dependency, right-click a record and click **Delete**. **For IDMS only:** - To reverse the direction of the area read, right-click a record and click **Reverse Area Read**. - To reverse the direction of the set read, right-click the record and click **Reverse Set Read**.

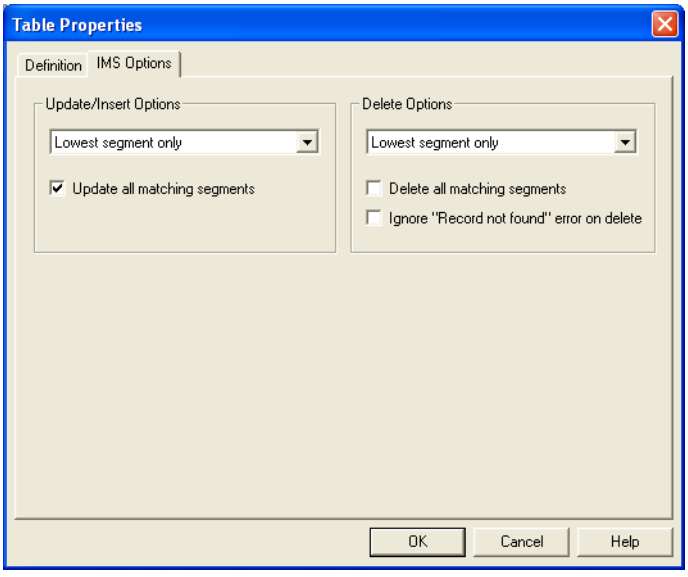
Option	Description
How do you want to handle multiple instances of selected records?	**New Row** - A new row is displayed or written to the target for every instance of the record or segment.
	Ignore - Second and subsequent instances of a record or segment displayed or written to the target.
	Array - The number of records or segments specified in the **Array** list are displayed or written to the target in a single row of output. PowerExchange populates the output row until it is full, and then either ignores subsequent records or segments or displays a new row with the overflow records or segments if the **New Row on Overflow** option is selected. For example, for a record with five instances, if you enter **3** in the **Array** list, PowerExchange builds two output rows. The first row contains an array of three instances, and the second row contains an array of two instances. **Note**: If a parent record or segment is set to **Array**, all child records or segments must be set to **Ignore**.

For an example of how to use these options to define a complex table, see "Example 3. Multiple-Record Data Map with a Complex Table" on page 26.

IMS Options tab:

The options on the IMS Options tab in the Table Properties dialog box enable you to define settings for segments in IMS complex tables.

The following figure shows the IMS Options tab:



The following table describes the options on the IMS Options tab that you can use to define IMS complex tables:

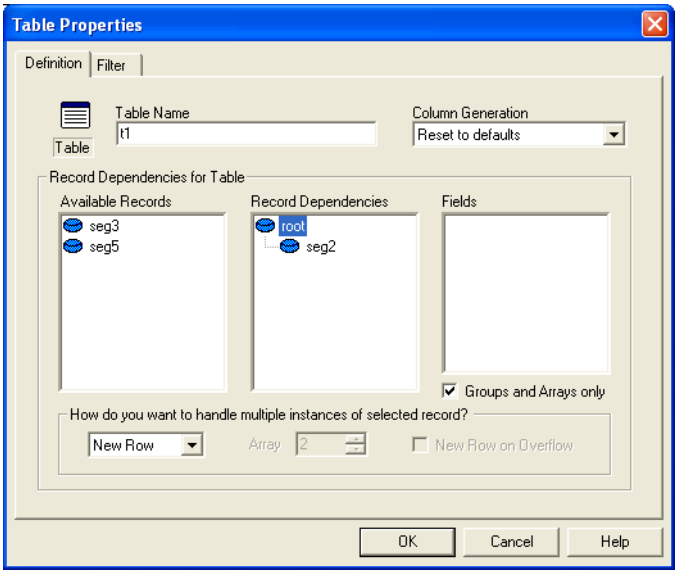
Option	Description
Update/Insert Options	Select one of the following options: - **Lowest segment only**. This is the default. Applies the action to the lowest segment only. - **All possible levels**. Applies the action to segment levels.
Update all matching segments	If selected, updates all non-unique matching segments. By default, this option is cleared.

Option	Description
Delete Options	Select one of the following options: - **Lowest segment only**. This is the default. Applies the delete action to the lowest segment only. - **All childless segments in hierarchy**. Applies the delete action to all childless segment levels.
Delete all matching segments	If selected, deletes all non-unique matching segments. By default, this option is cleared.
Ignore record not found error on delete	If selected, ignores all “Record not found” errors that are generated by the delete action. By default, this option is cleared.

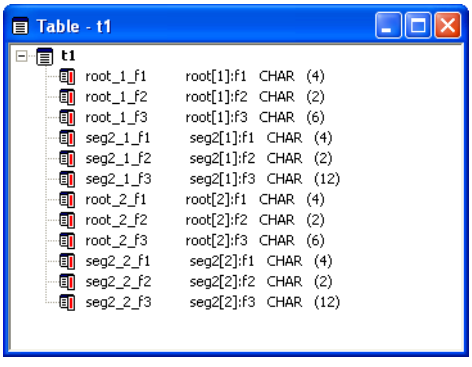
Examples. Complex Table Output

The following example illustrates a complex table that is defined with two segments that have a parent and child relationship.

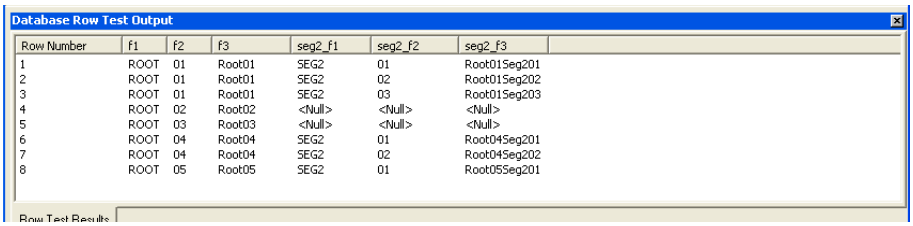
In the Table Properties dialog box, the table is defined as follows:



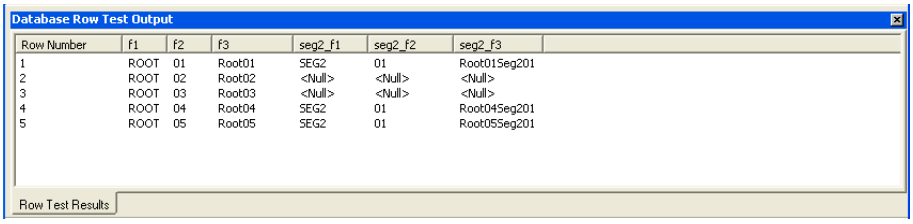
The **t1** table definition is displayed as follows:



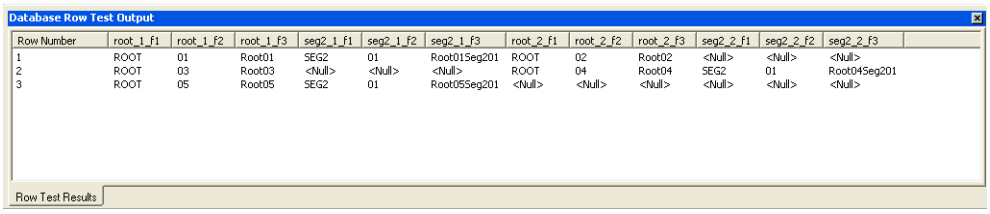
The following example shows the output for the **t1** complex table when the **root** and **seg2** segments are defined as **New Row**:



The following example shows the output for the **t1** complex table when the **root** segment is defined as **New Row** and the **seg2** segment is defined as **Ignore**:



The following example shows the output for the **t1** complex table when the **root** segment is defined as **Array** with and array size of 2, and the **seg2** segment is defined as **Ignore**:



Creating Data Maps - Examples

This section provides step-by-step examples for creating the following types of data maps:

- A single-record data map
- A single-record data map that contains arrays
- A multiple-record data map with complex tables

Example 1. Single-Record Data Map

This example illustrates how to create a data map by performing the following tasks:

1. Optionally add a resource configuration to define the location of your data maps, data files, and other resources.
2. Select an access method for, and associate a data file with, a data map.
3. Add a record to a data map.
4. Add fields to a record.
5. Test your data source and data map.

In this example, you use the sample demo1.dat file to create the demo.map1 data map.

Before You Begin

As part of the installation process, PowerExchange creates an examples subdirectory in the installation directory. For example:

```
C:\Informatica\PowerExchangen.n.n\examples
```

In this directory, find the demo1.dat file. Open it with a text editor, such as Notepad. Each record in this file contains three fields.

The data in this file is the basis for this example.

If you cannot find the demo1.dat file, create it with the following text:

```
10,Mickey Mouse,M
20,Shirley Temple,F
30,John Wayne,M
40,Donald Duck,M
50,Greta Garbo,F
```

Step 1. Select the PowerExchange Resource Configuration

To select a resource configuration:

- If you have defined multiple resource configurations, you can switch to the resource configuration that you want to use for the data map. For more information, see the "Current Configuration Tab" on page 8.

Step 2. Add a Data Map

In this example, you create the demo.map1 data map by using the sample demo1.dat file.

To add the demo.map1 data map:

1. Choose one of the following methods to add the data map:
   - On the toolbar, click the **Add Data Map** icon.
   - On the **Add** menu, click **Data Map**.
   - Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - In the **Schema Name** box, enter demo.
   - In the **Data Map Name** box, enter map1.
   - In the **Access Method** list, click **SEQ**.
   - Clear the **Import Record Definitions** option. You specify record definitions later.
3. Click **Next**.
4. In the SEQ Access Method dialog box, perform the following actions:
   - In the **File Name** field, browse to the examples directory and select demo1.dat. By default, the examples directory is in the PowerExchange installation directory, for example:

     ```
     C:\Informatica\PowerExchangen.n.n\examples\demo1.dat
     ```

   - Select the **Field Separator** option and enter the comma (,) separator character. If you cannot enter the separator character through the keyboard, enter its hexadecimal representation as: x'*nn*'. Because the data file, demo1.dat, is a character-separated file, you are not prompted for the separator character.
5. Click **Finish**.

   On the Data Map tab, the demo.map1 data map appears.

Step 3. Add a Record

In this step, you add the record1 record to the demo.map1 data map.

To add the record1 record:

1. On the Data Map tab, right-click demo.map1 and click **Add Record**.
2. In the **Record Name** box on the Add Record dialog box, enter record1.
3. Click **OK**.

   The Record window displays record1.

Step 4. Add Fields to the Record

In this step, you add the seqno, name, and gender fields to the record1 record.

To add the seqno field:

1. On the Data Map tab, click **record1**. Then click **Add > Field**.
2. In the Add Field dialog box, enter the following information.
   - In the **Field Name** box, enter seqno.
   - In the **Field Type** list, click **NUMCHAR**.
   - In the **Precision** list, enter 2.
   - In the **Scale** list, enter 0.
   - In the **Length** list, enter 2. The **Length** field indicates the maximum size of the input, allowing for leading blanks or other characters. The length value must be equal to or greater than the precision value.
3. Click **OK**.

   The Record window displays the seqno field for record1.

To add the name field:

1. In the Record window or on the Data Map tab, click **record1**. Then click **Add > Field**.
2. In the Add Field dialog box, enter the following information.
   - In the **Field Name** box, enter name.
   - In the **Field Type** list, click **CHAR**.
   - In the **Length** list, enter 20.
3. Click **OK**.

   The Record window displays the name field.

To add the gender field:

1. On the Data Map tab, click **record1**. Then, click **Add > Field**.
2. In the Add Field dialog box, enter the following information.
   - In the **Field Name** box, enter gender.
   - In the **Field Type** list, click **CHAR**.
   - In the **Length** list, enter 1.
3. Click **OK**.

   The Record window displays the gender field.

Step 5. Add a Table

In this step, you add the table1 table to the demo.map1 data map. The data from record1 makes up a row in table1.

To add a table:

1. On the Data Map tab, right-click **demo.map1** and click **Add Table**.
2. In the Add Table dialog box, perform the following actions:
   - In the **Table Name** box, enter table1.
   - In the **Record Dependencies** box, verify that the **record1** record appears.
   - Select the **Groups and Arrays only** option. This option causes the fields defined as a group field, or any array, to be displayed. If you clear this option, all fields in the record are displayed.
   - In the **How do you want to handle multiple instances of selected records?** list, click **New Row**, which causes a new row to be created for every instance of the record. For more information about the other options, see "Example 3. Multiple-Record Data Map with a Complex Table" on page 26.
3. Click **OK**.

   In the Table window, table1 appears, which contains all the fields that you added to record1.

Step 6. Test the Data Map

In this step, you perform a database row test on the demo.map1 data map. Rather than connecting to a remote PowerExchange Listener, access the data map locally.

For more information abut the database row test feature, see "Using Database Row Test" on page 141.

To test the demo.map1 data map:

1. In the Table window, select **table1** to display all table columns in the row test.

   Note: To display specific columns, select these columns in the Table window.
2. Click **File > Database Row Test**.
3. In the message box that prompts you to send the data map to a remote location, click **Yes**.
4. In the Data Map Remote Node dialog box, click **local** in the **Location** list.
5. Click **OK**.
6. In the Database Row Test dialog box, perform the following actions:
   - In the **DB Type** list, click **NRDB**. This field identifies the DB type of your data source.
   - In the **Fetch** list, click **Data**. This field identifies the type of data that you want to retrieve.
   - View the **SQL Statement** field, which displays the SQL generated by PowerExchange.
7. Click **Go**.
8. Close the Database Row Test Output window.
9. Close the Database Row Test dialog box.
10. To save the data map, click **File > Save**.
11. Click **File > Close Resource**.

    On the Resources tab, the demo.map1 data map appears in the **Data Maps** folder.

Example 2. Single-Record Data Map with an Array

In this example, you create a single-record data map from a file that contains multiple record types. This example demonstrates how to set options for a table to control how PowerExchange displays output from records that contain arrays. For more information, see “Defining Arrays” on page 12.

Before You Begin

In this example, you use the sample demo.map2 and demo2.dat files. In the **examples** directory, find the demo2.dat file. If you cannot find the demo2.dat file, create it with the following data:

```
10,Mickey Mouse,M,3,apple,orange,pear
20,Shirley Temple,F,1,raspberry
20,John Wayne,M,2,pansy,daisy
10,Donald Duck,M,0
30,Goofy,M,1,fox
10,Greta Garbo,F,3,dog,cat,rabbit
20,Ronald Reagan,M,2,horse,pony
20,Bette Midler,F,1,wolf
```

Step 1. Associate a Data File with a Data Map

Use the following procedure to associate a data file with a data map.

To associate a data file with a data map:

1. On the Resources tab, double-click **demo.map2** to open the data map.
2. On the Data Map tab, right-click **demo.map2** and click **Properties**.

   The Data Map Properties dialog box appears.
3. On the SEQ Access Method tab, click the **Browse** button next to the **File Name** box to browse to the demo2.dat file in the **examples** directory.
4. Click **OK**.
5. Click **OK**.

Step 2. Filter the Data

You can use the **Record ID Values** field in the Field Properties dialog box to filter the data that is displayed in a database row test. Record IDs provide a filtering mechanism at run-time, eliminating the need for SQL statements to select the desired records.

To exclude a data record:

1. On the Data Maps tab, click **record1** to displays its fields.
2. In the Record window, right-click the **rectype** field and click **Properties**.

   The Field Properties dialog box appears.
3. In the **Record ID Values** list, click the = or <> button to select the operators for filtering the data. 10 and 20, combined with the operator setting determine which data is displayed:
   - Click the = button to display records that have a record ID value equal to either 10 or 20. In the Database Row Test Output window, the **Goofy** record is not displayed because its record ID is not equal to 10 or 20.
   - Click the <> button to display records that have a record ID value that is greater than or less than (but not equal to) 10 or 20. In the Database Row Test Output window, the Goofy record is displayed because its record ID is greater than 20.

   Note: The data map stores the record IDs but not the data. Record ID fields might not be displayed in the column view of the data. Record ID filtering cannot be used on WRITE.

4. Click **OK**.

Step 3. Define a New Row for Each Element in an Array

This step demonstrates how to output a new row for each element in an array.

To define a new row for each element in an array:

1. In the Data Map tab, right-click the **row_out** table and click **Properties**.

   The Table Properties dialog box appears.

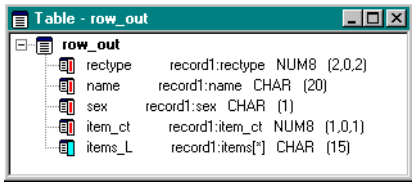
2. To display only fields defined as a group field or as an array in the **Fields** list, select the **Groups and Arrays only** option.

3. In the **Fields** list, select the **items** field. In this list, you can select only fields that are defined as arrays.

4. In the **How do you want to handle multiple instances of selected records?** list, click **New Row**. This option causes each occurrence of the data in the **items** field to be displayed as a new row. For information about the **Ignore** and **Array** options, see "Defining Arrays" on page 12.

5. Click **OK**.

6. On the Data Map tab, double click **row_out** table to open the Table Properties dialog box. The items_L column is displayed in turquoise, which indicates that it is part of an array:



Step 4. Test the Data Map

In this step, you perform a database row test on the demo.map2 data map. Rather than connecting to a remote PowerExchange Listener, you access the data map locally.

To test the demo.map2 data map:

1. On the Data Map tab, click **row_out**. Then click **File > Database Row Test**.

2. In the message box that prompts you to send the data map to a remote location, click **Yes**.

3. In the Data Map Remote Node dialog box, click **local** in the **Location** list.

4. Click **OK**.

5. In the Database Row Test dialog box, perform the following actions:

   - In the **DB Type** list, click **NRDB**.
   - In the **Fetch** list, click **Data**.

6. Click **Go**.

The Database Row Test Output window displays multiple output rows from a single input record, depending on the number of elements in the array. For example, because the Mickey Mouse record contains three items, the output displays three rows for that record as shown in the following figure:

Row Number	rec...	name	sex	ite...	items_L
1	10	Mickey Mouse	M	3	apple
2	10	Mickey Mouse	M	3	orange
3	10	Mickey Mouse	M	3	pear
4	20	Shirley Temple	F	1	raspberry
5	20	John Wayne	M	2	pansy
6	20	John Wayne	M	2	daisy
7	10	Donald Duck	M	0	<Null>
8	10	Greta Garbo	F	3	dog
9	10	Greta Garbo	F	3	cat
10	10	Greta Garbo	F	3	rabbit

7. Close the Database Row Test Output window.
8. Close the Database Row Test dialog box.

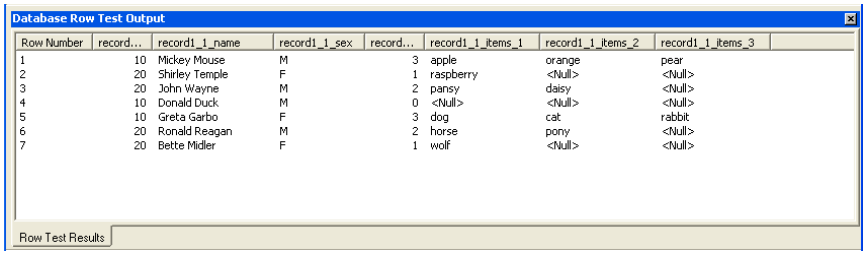
Step 5. Define a Single Row for All Elements in an Array

This step demonstrates how to output a single row for a record that contains an array. All elements in the array are displayed in a single row.

To define a single-row display for a record that contains an array:

1. In the Table Properties dialog box for the **row_out** table, perform the following actions:
   - In the **Fields** list, clear the **items** field.
   - In the **How do you want to handle multiple instances of selected records?** list, click **Array**.
   - In the **Array** list box, select **1**.
   - Select the **New Row on Overflow** option. PowerExchange populates the output row until it is full and then displays a new row with the overflow records.
2. Click **OK**.
3. Click **File > Database Row Test**.
4. In the message box that prompts you to send the data map to a remote location, click **Yes**.
5. In the Data Map Remote Node dialog box, click **local** in the **Location** list.
6. Click **OK**.
7. In the Database Row Test dialog box, perform the following actions:
   - In the **DB Type** list, click **NRDB**.
   - In the **Fetch** list, click **Data**.
8. Click **Go**.

   The Database Row Test Output window displays seven rows, one for each record. All elements of the array in a record appear in a single row, as shown in the following figure:



Row Number	record...	record1_1_name	record1_1_sex	record...	record1_1_items_1	record1_1_items_2	record1_1_items_3
1	10	Mickey Mouse	M	3	apple	orange	pear
2	20	Shirley Temple	F	1	raspberry	<Null>	<Null>
3	20	John Wayne	M	2	pansy	daisy	<Null>
4	10	Donald Duck	M	0	<Null>	<Null>	<Null>
5	10	Greta Garbo	F	3	dog	cat	rabbit
6	20	Ronald Reagan	M	2	horse	pony	<Null>
7	20	Bette Midler	F	1	wolf	<Null>	<Null>

Example 3. Multiple-Record Data Map with a Complex Table

In this example, you define a complex table with a hierarchical relationship between records of different types.

This example demonstrates the following concepts:

- A data map table can have multiple data map records associated with it.
- You can define a hierarchical relationship between the records.

Before You Begin

This example uses the sample demo3.dat and demo.map3 files in the **examples** directory. Verify that these files exist in this directory.

If you cannot find the demo3.dat file, create it with the following text:

```
1,Mickey Mouse
2,Disneyland
2,Mouse Hole
1,Shirley Temple
2,Hollywood
1,John Wayne
1,Donald Duck
2,Disneyworld
1,Ronald Reagan
2,White House
2,Hollywood
```

The demo.map3 data map contains the rec1 and rec2 records.

Step 1. Associate the Data File with the Data Map

Use the following procedure to associate the data file with the data map.

To associate the data file with the data map:

1. In the Resource Explorer, double-click the **demo.map3** data map to open the data map.
2. On the Data Map tab, right-click **demo.map3** and click **Properties**.

   The Data Map Properties dialog box appears.
3. On the SEQ Access Method tab, click the **Browse** button next to the **File Name** box to browse to the demo3.dat file in the **examples** directory.
4. Click **OK**.
5. Click **OK**.

Step 2. View the Records in the Data Map

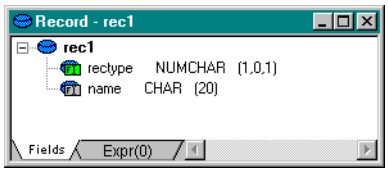
This step shows how record IDs are used to identify multiple records in a sequential flat file.

The first character of each line in the demo3.dat file, which the demo.map3 data map defines as the rectype field, distinguishes the rec1 record from the rec2 record.

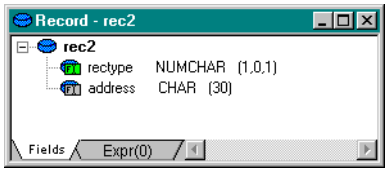
To view the records in the demo.map3 data map:

1. On the Resources tab, double-click **demo.map3** to open the data map.

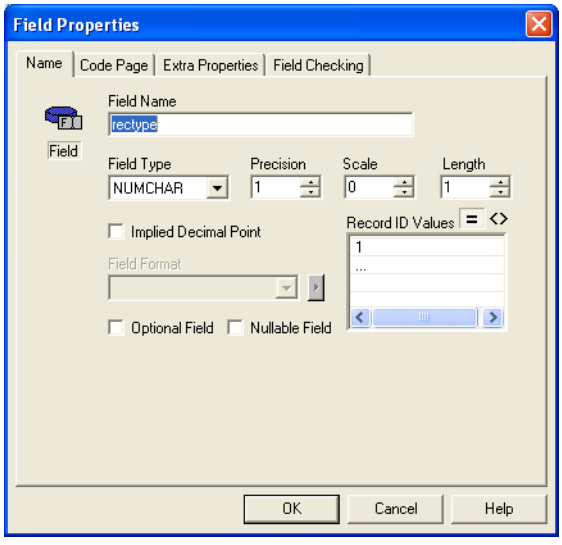
2. On the Data Map tab, click the **rec1** record to view its fields:



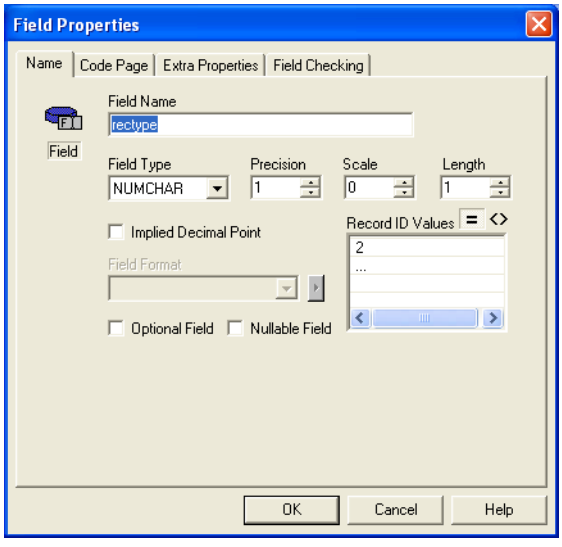
3. On the Data Map tab, click the **rec2** record to view its fields:



4. In the Record - rec1 window, double-click the **rectype** field to display its properties. The **Record ID Values** box indicates that the record ID for the rectype field is 1.



5. In the Record - rec2 window, double-click the **rectype** field. The **Record ID Values** box indicates that the record ID for the rectype field is 2.



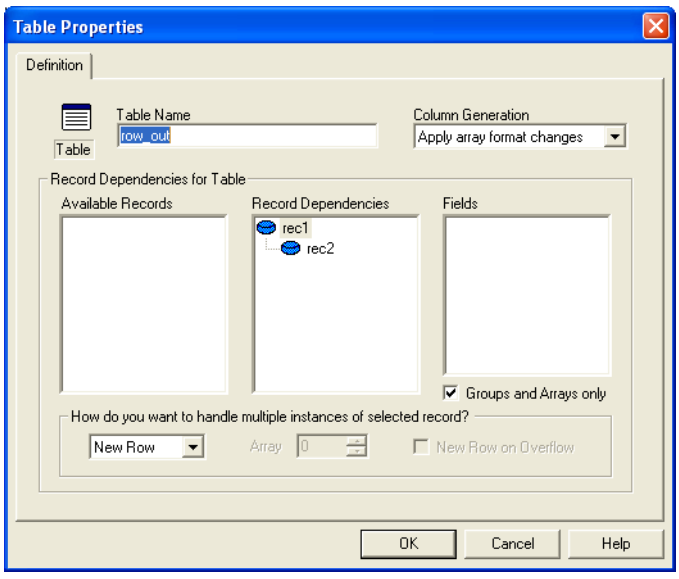
Step 3. Define a Hierarchy of Records

This step shows how to define a hierarchical relationship between the rec1 and rec2 records, which reflects the actual hierarchy in the data source.

To define a hierarchy:

1. On the Data Map tab, right-click the **row_out** table and click **Properties**.

   View the hierarchy between the rec1 and rec2 records in the **Record Dependencies** list:



2. To delete this hierarchy, right-click **rec2** and click **Delete**. This action moves the **rec2** record to the **Available Records** list.

3. In the **Record Dependencies** list, right-click **rec1** and click **Delete**. This action moves the **rec1** record to the **Available Records** list.

4. To redefine the hierarchy, in the **Available Records** list, right-click **rec1** and click **Add Record**. This action moves the **rec1** record to the **Record Dependencies** list.

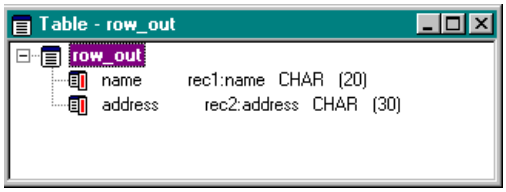
   For this record, the **How do you want to handle multiple instances of selected record?** option is set to **New Row**.

5. In the **Available Records** list, right-click **rec2** and click **Add Record as Child**. This action moves the **rec2** record to the **Record Dependencies** list as a child record of the **rec1** parent record.

   For this record, the **How do you want to handle multiple instances of selected record?** option is set to **New Row**.

6. Click **OK**.

   The following table is generated, which indicates that the **name** column is sourced from the **name** field in the **rec1** record, and the **address** column is sourced from the **address** field in the **rec2** record:



   The field name for each table column is prefixed with the source record name to create a unique reference for the field.

Step 4. Test the Data Map

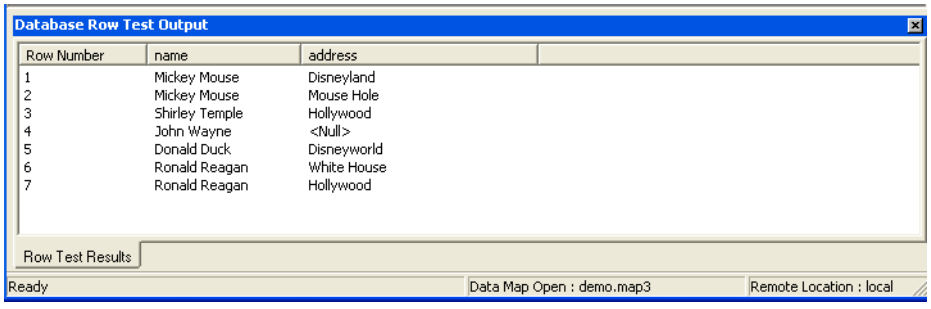
In this step, you perform a database row test on the demo.map3 data map. Rather than connecting to a remote PowerExchange Listener, you access the data map locally.

For more information abut the database row test feature, see “Using Database Row Test” on page 141.

To test the demo.map3 data map:

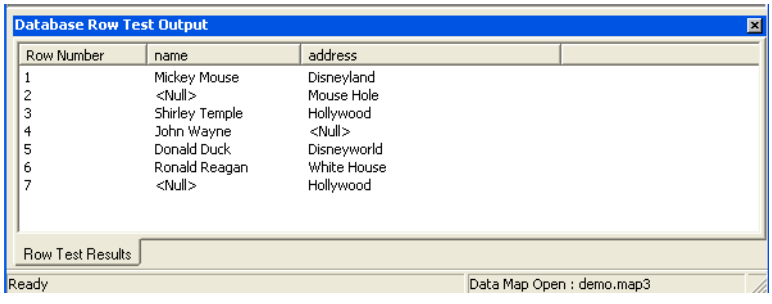
1. On the Data Map tab, click **row_out**. Then click **File > Database Row Test**.
2. In the message box that prompts you to send the data map to a remote location, click **Yes**.
3. In the Data Map Remote Node dialog box, click **local** in the **Location** list.
4. Click **OK**.
5. In the Database Row Test dialog box, perform the following actions:
   - In the **DB Type** list, click **NRDB**.
   - In the **Fetch** list, click **Data**.
6. Click **Go**.

   In the Database Row Test Output window, the **name** column is populated from the **rec1** record, and the **address** column is populated from the **rec2** record:



   Because the **rec1** record is the parent of the **rec2** record, row number 2 displays **Mickey Mouse** in the **name** column.

   However, if you change the hierarchy in the Table Properties dialog box for the **row_out** table to make the **rec2** record a sibling of the **rec1** record, the output in the Database Row Test Output window changes:



   Rows number 2 and 7 display **<Null>** in the name column for the following reasons:

   - No hierarchical relationship exists between the **rec1** and **rec2** records.
   - The **rec2** records for **Mouse Hole** and **Hollywood** do not have associated **rec1** records in the data file.

   Similarly, row number 4 displays **<Null>** in the address column because the **rec1** record for **John Wayne** does not have an associated **rec2** record in the data file.

The entries related to output rows number 2, 4, and 7 are highlighted in the following data file:

```
1,Mickey Mouse
2,Disneyland
2,Mouse Hole
1,Shirley Temple
2,Hollywood
1,John Wayne
1,Donald Duck
2,Disneyworld
1,Ronald Reagan
2,White House
2,Hollywood
```

Creating Adabas Data Maps

The first step to accessing Adabas through PowerExchange is to create a data map with PowerExchange Navigator. Adabas provides several data definitions that you use to create data maps:

- **DDM and PREDICT.** DDM and Predict are the best options as they provide long name and scale information.

  To import these types it is necessary to know the FDIC database ID and file number. Given this information it is possible to browse for the required data structure (wildcards are supported) or specify the name directly if known.

  Note: If you import DDM metadata for Adabas databases on MVS that contain wide character datatypes, PowerExchange does not automatically select the **Wide Char** option on the Code Page tab for these fields. You must either import Predict or FDT metadata or manually select the **Wide Char** option in the field properties for all wide character datatypes.
- **FDT.** The Field Definition Table (FDT) is an Adabas construct that contains the database definition. It also enables validation of Adabas related information such as field names, array sizes, and override lengths.Importing the Field Definition Table (FDT) ensures that the latest definition is selected but provides only short names and no scale information. Once imported the FDT is also cached for subsequent sessions. It can be refreshed through the FDT options. The caching of FDT allows certain validity checks to be made prior to run time.
- **DDM OPEN SYS.** DDM OPEN SYS (or Open System) is used to support Adabas databases on Windows and UNIX.
- **ADACMP.** ADACMP cards can be imported directly from the host as is done for the other file types or from a local file on the machine.
- **TEXT.** TEXT is a delimited format file which uses an Informatica internal format.

Note: You can also import COBOL and PL/1 copybooks. As these do not contain any Adabas field information the resulting PowerExchange field definitions must be associated with Adabas field definitions in the Field Properties dialog box.

You can subdivide, or redefine, any PowerExchange field mapped to an Adabas field. To subdivide a field, defining it as a group field with subsidiary fields that re-map the group.

To create an Adabas data map:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - Select the **ADABAS** access method.
   - Enter values for **Schema Name** and **Data Map Name**.

   By default, the **Import Record Definitions** and **Import Key Fields/FDT** options are selected.
3. Click **Next**.

4. In the ADABAS Access Method dialog box, enter the following information:

Field	Description
Database ID	The Adabas database number.
File Number	The file number in the Adabas database.
Optimization Level	Specifies how PowerExchange uses key fields. Select one of the following options: - **Full.** Use Adabas keys (Descriptors) when a SQL WHERE clause is present. - **OFF.** Do not use Adabas keys (Descriptors). - **Ranges Only.** Use Adabas Keys (Descriptors) only when the key value is in the range given, when a SQL WHERE clause is present. The default is Ranges Only.
Encoding	Specify one of the following data encoding schemes: - **ASCII HIEND.** ASCII high endian - **ASCII LOEND.** ASCII little endian - **Default.** Uses the appropriate value based on the platform - **EBCDIC.** EBCDIC The default is Default.
Codepage	The code page that describes the character set for the character data in the database. PowerExchange Navigator enables this option if you chose a value other than Default for Encoding. Otherwise, PowerExchange uses the default code page of the platform. For more information about code pages, see “Data Map Code Pages” on page 72. The default is Default.
Wide Character Codepage	The code page for Adabas W format fields, also called wide character datatypes. Default uses UTF-8 for UNIX and Windows platforms and the platform default for MVS. The default is Default.

The Database ID and File Number normally equate to the Database ID and File Number of the expected source of the Adabas data. PowerExchange provides the flexibility at this point to set the Database ID to zero.

If the actual Database ID is zero, then PowerExchange accesses the data correctly. Setting Database ID to zero enables the database to be overridden at runtime using one of the following methods:

- ADABAS_DEFAULT_DBID statement in the DBMOVER configuration file
- Database Id Override option in the session properties in PowerCenter
- Override File Name in the ODBC escape sequence DTLDSN or ODBC parameter DBQUAL1.

Tip: Setting the Database ID to zero and overriding the value in PowerCenter makes migrating data maps from one Adabas environment to another easier.

5. Click **Finish**.

   The Adabas FDT Import dialog box appears. Enter the node name specified in dbmover.cfg on Windows for the Adabas source system. If the source system is MVS and PowerExchange is configured for security, enter a valid MVS user ID and password in the **User ID** and **Password** fields. If you use Adabas file security, enter the password in the **File Password** field.

6. Click **OK**.

7. In the Import Copybook - Source Details dialog box, enter the following information:

Field	Description
Source	The location of the copybook or metadata information. Select **Local** if the metadata is available on this Windows machine or **Remote** if the metadata is remote from this Windows machine.
Source Type	The formats for the copybook or metadata, which is one of the following values: - **ADACMP.** File created using the ADACMP utility - **COBOL.** COBOL copybook - **DDM.** Natural Data Definition Module (DDM) - **DDM OPEN SYS.** Natural Data Definition Module (DDM) for UNIX and Windows systems - **FDT.** Adabas Field Definition Table (FDT) for the database - **PL/1.** PL/I copybook - **PREDICT.** Predict data dictionary - **TEXT.** Text file. Only available if you select Local. The default is DDM.
Column Range	Defines the start and end columns that should be inspected. Typically this should be set to 7-72 for COBOL copybooks,1-72 for PL/1 copybooks, and 1-80 for the ADABAS copybooks. Available range is 1-999.
FDIC File Details	When Type is **DDM** or **PREDICT**, specify the database number and file number for the data dictionary.
Natural User Library Path	When Type is **DDM OPEN SYS**, specify the path for the Natural user library that contains the DDM for the database.

8. Click **Next**.

9. In the Import Copybook - Remote Details dialog box, enter the following information:

Setting	Description
File Password	Adabas file password, if required.
Location	Target location as defined in the dbmover.cfg file.
UserId/Password	If PowerExchange is configured for security, a valid MVS user ID and password.
Save File Locally As:	File into which the copybook or metadata is saved locally.
Name	Enter Adabas name or select from the Browse List.
Name Browse	To generate a selection list leave the Name blank and click Name Browse. Click the selected name from the list which then appears in the Name field.
Preview	Click this button to see the Adabas layout of the selected Name.

If you select **DDM**, **DMM OPEN SYS**, or **PREDICT**, enter a name in the Name field. The Name Browse button provides the opportunity to browse for the required file and the Preview button enables you to view it.

10. Click **Next**.

11. In the Import Copybook - Configuration Details dialog box, select options.

12. Click **Finish**.

13. The Record Definition dialog box displays with the Adabas record name in the **Name** field.

14. Click **OK**. The data map definition is now complete.

15. Close and save the new data map.

Step 1. Import the Adabas FDT

When you open an Adabas data map that has not had the FDT imported, a message box is displayed asking you if you wish to import the FDT details now. If you click **No**, the data map is displayed. If you want to import the FDT later, click **Resource > FDT**. Alternatively, right-click the data map and click **FDT**.

To import the Adabas database FDT:

1. When you open the data map, click **Yes** to the prompt to import the FDT.
2. The Adabas FDT Import dialog box displays. Enter the Location of the FDT information. If required, enter values for User ID, Password, and File Password.
3. Click **OK**. The Adabas FDT dialog box displays showing the FDT information.

   Click **Refresh FDT** to refresh the FDT information.
4. Click **OK**.

Step 2. Import Adabas Keys

You can import Adabas keys when importing FDT metadata.

To import Adabas keys for FDT data maps:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, enter values for **Schema Name** and **Data Map Name**.
3. Select **Import Record Definitions** and **Import Key Fields/FDT**. This populates a list on the Keys tab of the Data Map Properties dialog box.
4. For each mapped table you can define a key. Highlight the table and select the Properties right-menu option.
5. Click the Keys tab.
6. The Primary Key list holds valid keys for the table that has been defined. The validity of a key is determined by checking that the table contains all the fields that are defined in the record as having the Adabas field names needed by the descriptor. If any of these fields have been mapped into groups, then all members of the group must also be present. Fields in vertical arrays are not valid.

   The Key Details list shows the key that has been selected for the table from the Primary Key list.

Testing an Adabas Data Map

To test the data map, use the database row test feature. The database row test options retrieves data from the selected Adabas database. For more information about testing Adabas data maps, see “Using Database Row Test” on page 141.

Creating Datacom Data Maps

The first step to accessing Datacom through PowerExchange is to create a data map with the PowerExchange Navigator.

Data maps create a relational view of the nonrelational data. These definitions are then used to access the nonrelational data.

The recommended method of obtaining the Datacom definition is for the Datacom copybook to be imported automatically, as described in the following procedure.

There is also a manual method that allows the use of COBOL and PL/1. For more information about importing a COBOL or PL/I copybook, see "Importing Copybooks" on page 75.

To create a Datacom data map:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, select **DATACOM** in the **Access Method**.
3. Enter values for **Schema Name** and **Data Map Name**.

   These names do not need to relate to any Datacom structures.
4. Select the **Import Record Definitions** option to enable the definitions to be imported from Datacom.
5. Enter the appropriate values in **Database Name** and **Database ID**.

   Note: Both the **Database Name** and **Database ID** are mandatory. If the data map is being built from scratch, and if the Import Record Definition was not selected on the previous screen, then enter the appropriate value in **Record Size**.
6. Click **Finish**.
7. The Import Copybook wizard appears. Select **Remote**.
8. Select **DATACOM**, **COBOL**, or **PL/I** for the copybook source **Type**.
9. Click **Next**.
10. In the Import Copybook dialog box, enter the following information.
    - Enter two user IDs and passwords. as required by the configuration of PowerExchange and Datacom. The requirement for the first user ID and password is dependent upon the SECURITY parameter in the DBMOVER configuration file. For more information about PowerExchange security, see the *PowerExchange Reference Manual*.
    - The Dictionary UserID and Password are required if Datacom requires them. Enter the Datacom Table Name, Location, which is a node in the dbmover.cfg file on the PC, and a name for the generated map to be held. If overwriting an existing one, the Browse button can be used to find the required folder and file.
    - If a Copybook is to be imported with the metadata, select **Copybook** and then select either **COBOL** or **PL/I**.

    Note: The Table Status and Key and Elements fields are disabled at this time.
11. Click **Next** or **Finish**.

    If you select **Next**, the Configuration Details dialog box displays. Click **Finish** to display the summary information. Then click **OK**.

    If you click **Finish**, the Configuration Details dialog box is skipped.
12. If Record Definition dialog box displays, click **Apply** to import the rest of the copybook.
13. Click **OK**.

    The Datacom Import dialog box appears after all records have been imported.

14. Close the Datacom Import dialog box.

Testing a Datacom Data Map

To test the data map, use the database row test feature. The database row test features retrieves data from the selected Datacom database. For more information about testing a Datacom data map, see “Using Database Row Test” on page 141.

Datacom Record Properties Dialog

To view Datacom-specific record properties, right-click the record in data map and click **Properties**.

Edit the attributes of the record definition from this screen.

Elements Tab

An element is the unit of transfer used between applications issuing Datacom commands and Datacom. It consists of one or more contiguous columns, which are FIELD entity occurrences. An element should contain only those columns that an application program uses at execution time. Each table requires one to 255, elements.

To display the element information:

- Click the Elements tab. The Record Properties dialog box is displayed.

  The Elements Record Properties dialog box contains the following fields:

Field	Description
Name	Name of the record element to be used.
Position	Position offset from the start of the record.
Length	Length of the element.

To add an element:

1. Click the Elements tab for the record on the data map.
2. Click the **Add** icon.
3. Enter a name for the element in the **Name** box, and then enter values for the **Position** and **Length** fields.
4. Click **OK**.

   The new element details display in the Record Properties dialog box.

To delete an element:

1. Click the Element tab for the record on the data map.
2. Highlight the desired Element entry you want to delete.
3. Click the **Delete** icon.

Keys Tab

Keys are structures used to optimize data access or order data retrieval. A key is composed of columns that can be contiguous or non-contiguous and in any sequence. Each column in a key can be ascending or descending in value. Each key can be up to 180 characters long. Define up to 99 keys for each table or 999 keys for each database. Any key can be defined as unique, that is, requiring that each row in the table have a unique value for the key.

All tables must have a Master Key and a Native Key defined. The Master Key functions as any other key, but it can be defined as updatable or non-updatable. The Native Key dictates the physical sequence in which the data is stored. The Native Key can be the same as the Master Key.

Click the Keys tab. The Keys Record Properties dialog box displays with the long and short names for all of the keys.

To add a key:

1. Click the Keys tab for the record on the data map.
2. Click the **Add** icon.
3. The Key Properties dialog box displays. Enter key name values in the **Long Key Name** and **Short Key Name** fields. Select any appropriate key attributes.
4. Click the Segment Details tab to assign the new segment details. Enter the segment name in the **Name** field. Enter values for the **Position**, **Length**, **Order**, and **Sensitivity** fields.
5. Click **OK** to display the new structure.
6. Click **OK**.

   The key appears in the Record Properties dialog box.

To delete a key:

1. Click the Keys tab for the record on the data map.
2. Highlight the key entry you want to delete.
3. Click the **Delete** icon.

Creating DB2 Data Maps

Because DB2 is a relational database, you do not need to create a data map for DB2 tables to allow PowerExchange to access them. However, some DB2 tables use single DB2 columns to store an array of fields in a format not necessarily consistent with the column type. For example, a CHAR column might contain a set number of packed data fields, or a VARCHAR column might contain a varying number of packed fields. When you create a DB2 data map for tables with these types of columns, PowerExchange allows you to manipulate the sub-fields in the columns using expression fields and expose them to downstream processing by PowerCenter.

After DB2 data maps are available they can be extended to make it easier to map columns that have been unloaded into flat files by standard utilities.

PowerExchange provides data maps for DB2 tables and for DB2 unload files. For DB2 unload files, PowerExchange supports the following IBM and BMC Software formats:

- DB2 for z/OS online REORG TABLESPACE utility with UNLOAD EXTERNAL
- DB2 for z/OS online UNLOAD utility
- DB2 for z/OS sample unload program DSNTIAUL
- BMC Software Unload Plus
- DB2 for Linux, UNIX, and Windows High Performance Unload utility

Note: When you create a DB2 unload data map and select **DSNUTILB UNLOAD** for the **Unload Type**, PowerExchange includes a field for the OBID in the record. If you create DB2 unload files specifying HEADER NONE in the DB2 UNLOAD control statements, select **UNDEFINED** for the **Unload Type**.

Creating a Data Map by Using the DB2 Catalog

Although this example illustrates DB2 for z/OS, you can use the same process to create data maps for DB2 for i5/OS and DB2 for Linux, UNIX, and Windows.

To create a data map for DB2:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - In the **Schema Name** box, enter db2.
   - In the **Data Map Name** box, enter map1.
   - In the **Access Method** list, click **DB2**.
   - Select the **Import Record Definitions** option.
3. In the DB2 Access Method dialog box, perform the following actions:
   - In the **DB Instance** box, enter the DB instance, such as DSN1.
   - In the **Table Name** field, enter the table name or click the browse button. In the DB2 Table Filter dialog box, enter filter criteria based on the creator or schema name, database name, or table name. Click **Next**. In the DB2 Table Filter dialog box, select a table and click **Finish**.
4. Click **Finish**.
5. In the Import Copybook - Source Details dialog box, perform the following actions:
   - In the **Source** area, select **Remote**.
   - In the **Type** list, click **DB2 Catalog**.
6. Click **Next**.
7. In the Import Copybook - Remote DB2 Catalog Details dialog box, perform the following actions:
   - If you want, change the import details. The input fields are populated from previous selections.
   - In the **Save File Locally As** box, enter a file name.
   - Click **Preview/Amend Columns** to select specific columns.
8. **Click Next**.
9. In the Import Copybook - Configuration Details dialog box, select which actions are to occur when loading the copybook.
10. Click **Finish**.
11. In the Import Copybook Information dialog box, click **OK**.
12. In the Record Definition dialog box, click **OK** for the table.

    Note: To suppress the Record Definition dialog box, clear the **Prompt on the record import** option in the Import Copybook - Configuration Details dialog box.

    The Copybook Message Log is displayed.
13. Close the DB2 Catalog Import window.
14. On the Data Map tab, click the imported record and table to view the fields and columns.

Test a DB2 Data Map

For more information about the database row test feature, see “Using Database Row Test” on page 141.

To test a DB2 data map:

1. On the Data Map tab, click a table. Then click **File > Database Row Test**.
2. In the message box that prompts you to send the data map to a remote location, click **Yes**.

3. In the Data Map Remote Node dialog box, click **local** in the **Location** list.
4. Click **OK**.
5. In the Database Row Test dialog box, perform the following actions:
   - In the **DB Type** list, click **NRDB**.
   - In the **Fetch** list, click **Data**.
6. Click **Go**.

   The Database Row Test Output window displays the output.
7. Close the Database Row Test Output window.
8. Close the Database Row Test dialog box.
9. To save the data map, click **File > Save**.
10. Click **File > Close Resource**.

Creating a Data Map by Using a DB2 Unload File

In addition to direct data mapping of DB2 tables, you can also map DB2 unload files.

The data map creation process is the same as for any other data source that you have mapped using PowerExchange. The following sections give an example of creating the data map and then testing it.

To create a DB2 data map with a DB2 unload file:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - In the **Schema Name** box, enter db2unload.
   - In the **Data Map Name** box, enter map1.
   - In the **Access Method** list, click **DB2UNLD**.
   - Select the **Import Record Definitions** option.
3. In the DB2UNLD Access Method dialog box, perform the following actions:

   In the **File Name** box, enter the name of the target DB2 unload file.

   In the **DB Instance** box, enter the DB instance, such as DSN1.

   In the **Table Name** field, enter the table name or click the browse button. In the DB2 Table Filter dialog box, enter filter criteria based on the creator or schema name, database name, or table name. Click **Next**. In the DB2 Table Filter dialog box, select a table and click **Finish**.

   In the **Unload Type** list, select the unload type. For bulk access to unload files, the following IBM and BMC formats are supported:
   - REORG UNLOAD
   - DSNTIAUL/BMC UNLOAD+
   - DSNUTILB UNLOAD
   - UNDEFINED. By default, this type is generated with null indicators, and count fields in the same position as in the REORG UNLOAD EXTERNAL type.

   Note: Instead of using these unload types, you can modify these statements to meet your specific requirements. You can move or delete these statements to correspond to user-defined unload formats, but PowerExchange interprets them only if they precede the data field to which they apply.
4. Click **Finish**.

5. In the Import Copybook - Source Details dialog box, perform the following actions:
   - In the **Source** area, select **Remote**.
   - In the **Type** list, click **DB2 Catalog**.
6. Click **Next**.
7. In the Import Copybook - Remote DB2 Catalog Details dialog box, perform the following actions:
   - In the **Save File Locally As** box, enter a file name.
   - In the **Null Indicator** box, enter the hexadecimal value of the null indicator that is used in the unload file that you are mapping.
   - Select the **Pad Variable** option if the variables are padded in the unload file that you are mapping. This option corresponds to the BMC Unload+ FIXEDVARCHAR.
   - Click **Preview/Amend Columns** to select specific columns.
8. Click **Next**.
9. In the Import Copybook - Configuration Details dialog box, select which actions are to occur when loading the copybook.
10. Click **Finish**.
11. In the Import Copybook Information dialog box, click **OK**.
12. In the Record Definition dialog box, click **OK** for the table.

    Note: To suppress the Record Definition dialog box, clear the **Prompt on the record import** option in the Import Copybook - Configuration Details dialog box.

    The Copybook Message Log is displayed.
13. Close the DB2 Catalog Import window.
14. On the Data Map tab, click the imported record and table to view the fields and columns.

Test a DB2 Unload File Data Map

Use the following procedure to test a DB2 data map that uses a DB2 unload file.

To test a DB2 data map that uses a DB2 unload file:

1. On the Data Map tab, click a table. Then click **File > Database Row Test**.
2. In the message box that prompts you to send the data map to a remote location, click **Yes**.
3. In the Data Map Remote Node dialog box, click **remote** in the **Location** list.
4. Click **OK**.
5. In the Database Row Test dialog box, perform the following actions:
   - In the **DB Type** list, click **NRDB**.
   - In the **Fetch** list, click **Data**.
6. Click **Go**.

   The Database Row Test Output window displays the output.
7. Close the Database Row Test Output window.
8. Close the Database Row Test dialog box.
9. To save the data map, click **File > Save**.
10. Click **File > Close Resource**.

Access Unload Data from Multiple Tables

Multi-table tablespaces result in a single unload file that contains data from several tables. Individual tables can be selected from the unload using the OBID of the table.

Two types of unload file are supported for this type of multiple table access:

- Type 1 REORG UNLOAD
  OBID is held in the fourth and fifth bytes of each record
- Type 3 DSNUTILB UNLOAD
  OBID is held in the first two bytes of each record

The process for creating a data map that can be used to access multiple tables in the same unload file is as follows:

1. A data map is created as normal.
2. The underlying tables are added.
3. The underlying table for each record type in the map is specified by entering the appropriate OBID into the record properties dialog box for the DTL__OBID field.

Example

The following example uses a Type 3 DSNUTILB UNLOAD file but the process is the same if you are using a Type 1 REORG UNLOAD file.

Scenario

- The unload file that is being mapped contains data from two underlying tables each with its own structure.
- Two tables are PWXTAB3 and PWXTAB4.
- The underlying table is specified by the OBID in the first two (for Type 3) bytes. The OBIDs for the tables are x'0004' and x'0005' respectively.

Procedure

Use the following procedure to perform a database row test on a DB2 data map that uses a DB2 unload file.

To test a DB2 data map that uses a DB2 unload file:

1. Map the table (PWXTAB3) in the normal way. See "Creating a Data Map by Using a DB2 Unload File" on page 38.
2. Right-click the DTL_OBID field and select the Properties menu item. Enter the OBID hex value of x'0004' into the Record ID Values box.
3. Click **OK**. The DTL_OBID field icon is green showing that the field has a Record ID filter set.
4. You can then import the next underlying table (PWXTAB4) and repeat steps 2 and 3 using an OBID value of x'0005'.

   You can now test that the OBID filter is working by performing a row test against one of the two tables. In this case, you row test the PWXTAB4 table.
5. Select table PWXTAB4 and click the **Row Test** icon.
6. Click **Go**.

Note: Only two rows from the unload file are returned. These rows have an OBID of x'0005'.

Amending the Record Layout

One of the advantages of DB2 data maps is that the record layout of the DB2 table can be enhanced by amending the data map.

To group these three fields together as a single field called MISC_DATA:

1. In the record display, select the MISC_1 field.
2. Right-click to select Properties.

   This displays the Field Properties dialog box.
3. Change the Field Type to GROUP and the name to MISC_DATA.

   You can then add fields to the GROUP field.
4. Right-click and choose **Add Field as Child**. Add the MISC_1, MISC_2, and MISC_3 fields as children.

Creating IDMS Data Maps

PowerExchange enables access to nonrelational data through SQL as though the data source was relational. The first step to accessing IDMS through PowerExchange is to create a data map with PowerExchange Navigator. Use PowerExchange Navigator to build the following components:

- **Data maps.** The relational to nonrelational definitions that are used to access IDMS data sources.
- **Personal metadata.** Parameters to browse metadata from remote databases.

Adding an IDMS Data Map

The IDMS copybook should be imported automatically. There is a manual option to allow for the use of COBOL and PL/1.

PowerExchange downloads metadata from MVS to create the data maps for IDMS. When running with a security level of 0 or 1, data sets are created on MVS prefixed with the user ID under which the PowerExchange Listener is running. If this user ID is not acceptable, use TEMPHLQ statement in the DBMOVER configuration file. For more information about PowerExchange security, see the *PowerExchange Reference Manual*.

To add an IDMS data map:

1. Click **Add** > **Data Map**.
2. In the Name dialog box, enter a schema name and a data map name and select the **IDMS** access method.
3. To import the record layout, select **Import Record Definitions** and click **Next**.

   The IDMS Access Method dialog box appears. This dialog box has the following fields:

Field Name	Description
Sub Schema Name	Required. Name of the sub-schema to be used for data retrieval.
DBName	Database name
Program Name	Program identification string
DBNode	Distributed Database System (DDS) node name
Dictionary Name	Name of the dictionary
Dictionary Node	DDS dictionary node name

Field Name	Description
Ready Mode	Select one of: - **Exclusive.** Prevents concurrent use by any others. - **Protected.** No updates are allowed by other processes until the run unit is complete. - **Shared.** Default. No locking or protection is activated. - Select one of: - **Retrieval.** Default. Data retrieval mode. - **Update.**Data update mode.
Ready All	If selected, ensures all areas are readied.

Note: The **Sub Schema Name** field is mandatory. All of the other fields are optional and are determined by the installation configuration and standards.

4. Click **Finish**.

5. The **Import Copybook** wizard opens automatically. Select **Remote** and **IDMS** in the **Type** field.

   COBOL and PL/I imports are allowed with IDMS. However, you must add the IDMS record navigation information manually. For more information, see "Add Owner and Set Names" on page 44. For more information about COBOL and PL/I copybook imports, see "Importing Copybooks" on page 75.

6. Click **Next**.

   The Import Copybook - Remote IDMS Details dialog box displays. This dialog box has the following fields:

Field	Description
Dictionary Name	Required. The name of the dictionary.
Location	The node name from the NODE statement in dbmover.cfg on Windows for the MVS system where the IDMS database resides.
Save File Locally As	The local file name for the IDMS metadata. Click **Browse** to search for the file.
DBName	The database name.

Note: IDMS parameters are determined by the installation configuration and standards.

Click **Advanced**. The IDMS Advanced Properties dialog box displays. Use the Advanced button to provide additional information for acquiring the IDMS metadata. This dialog box has the following fields:

Field Name	Description
Schema Name	Name of the schema to be used for data retrieval.
Schema Version	Version of the schema specified in Schema Name.
Dictionary UserID	User ID with access to the dictionary.
Dictionary Password	Password for the dictionary user ID.
Node Name	Distributed Database System (DDS) node name
Dictionary Node	DDS dictionary node name

If a user ID and password are required to access the dictionary, enter them on the IDMS Advanced Properties dialog box.

7. Click **OK** or **Cancel**.

   If you click **OK**, the PowerExchange Navigator communicates with the PowerExchange Listener on the MVS system specified by the node name in the **Location** field. The PowerExchange Listener then submits batch job IDMSMJCL to import the IDMS metadata. IDMSMJCL is a member in the RUNLIB library on that MVS system.

8. Click **Next** or **Finish**.

   If you click **Next**, the Import Copybook - Configuration Details dialog box displays.

   IDMS log-based change data capture works at a single record level. On this dialog box, two options determine the representation of data required. If this data map is to be used for change data capture, make sure the **Create Table on Each Record Imported** option is selected.

   If this data map is being set up for bulk data movement, relational representation of the IDMS data may be required. If so, check the **Create Tables for IDMS Hierarchical Paths** option.

9. Click **Finish**.

10. Click **OK**.

11. On the Record Definition dialog box, click **Apply** to import the rest of records.

12. Click **OK** for the COVERAGE record.

13. Close the IDMS Import dialog box.

Testing an IDMS Data Map

You can test a data map by using a database row test. A database row test accesses actual source data and displays it in table format. You can use a database row test to ensure that data can be retrieved from the source database.

For more information, see "Using Database Row Test" on page 141.

Changing IDMS Record Properties

You can change IDMS record properties after you have imported the IDMS metadata.

To change IDMS record properties:

1. Right-click the record and click **Properties**.

   The Record Properties dialog box displays. The Name tab of this dialog box has the following fields:

Field	Description
Record Name	PowerExchange record name in the data map.
IDMS Record Name	The internal IDMS record name as known to IDMS.
Area Name	The internal IDMS area name within which the IDMS record exists.
Record ID	Uniquely identifies each record type within the schema.
Location Mode	The technique that IDMS uses to physically store occurrences of the record type: - **CALC**. A method of determining the target page for storage of a record in the database. The target page is calculated by a randomizing routine executed against the CALC key in the record. Enables the Duplicates list. - **DIRECT**. Occurrences of the record are stored on or near a page specified by the user program at runtime. - **VIA**. Occurrences of the record in a specific set are stored relative to their owner. - **VSAM**. Specifies the record is native VSAM format. - **VSAM CALC**. Specifies the record is native VSAM format for which CALC access is required. Enables the VSAM Type and Duplicates lists and the CALC Element Names tab.

Field	Description
Duplicates	Enabled if CALC or VSAM CALC is selected from the Location list: - Not Allowed. IDMS does not store occurrences with duplicate CALC keys. - First. Logically positions record occurrences with a duplicate CALC key before the duplicate record already stored. - Last. Logically positions record occurrences with a duplicate CALC key after the duplicate record already stored. - By DBKey. Logically positions record occurrences with a duplicate CALC key according to the db-key.
VSAM Type	Identifies the record as a native VSAM and how the file that contains the record was defined. Enabled if VSAM or VSAM CALC is selected from the Location list: - Fixed NonSpanned. Specifies a fixed length record that cannot span VSAM control intervals. - Fixed Spanned. Specifies a fixed length record that can span VSAM control intervals. - Variable NonSpanned. Specifies a variable length record that cannot span VSAM control intervals. - Variable Spanned. Specifies a variable length record that can span VSAM control intervals.
Page Group	Maximum value is 32767.
Radix	Valid values are 0 to 12.
Compress	Records held in compressed format.
Variable	Records held as variable record length.

Click the Owner Records and Set Names tab. This tab has the following fields:

Field	Description
Owner Name	IDMS owner records of this record as defined within IDMS.
Set Name	Relevant IDMS set names.

Add Owner and Set Names

You can add owner and set names for the records in the display on the Owner Records and Set Names tab.

To add an owner and set name:

1. On the Owner Records and Set Names tab, click on the record.
2. Click the **Add Owner** icon.
3. The Owner Record and Set Details dialog box displays.

   The field values must be valid IDMS record names and set names and have a valid relationship with the record for which the properties are being added.
4. Click **OK**.

   The new details appear on the Owner Records and Set Names tab.

   Note: This process is not recommended for setting up an IDMS data map, but this process is available for flexibility. For information about the recommended process, see the import procedure described in "Adding an IDMS Data Map" on page 41.

   For more information about the objects on this tab, see the online help.

Delete an Owner and Set Name

Use the following procedure to delete an owner and set name.

To delete an owner and set name:

1. On the Owner Records and Set Names tab, click on the record.
2. Click the **Delete Owner** icon.

Calc Element Names Tab

The Calc Element Names tab appears if you select a location of **CALC** or **VSAM CALC** on the Name tab. This allows you to define one or more fields that are combined to form a single key for the record. The following table describes the fields in this dialog box:

Field	Description
Name	Name of the record element to be used.
Position	Position offset from the start of the record.
Length	Length of the element.

To add a calc element name:

1. Click the Calc Element Name tab for the record on the data map.
2. Click the **Add** icon. Enter the new CALC element name in the **Name** field and provide values in the **Position** and **Length** fields.
3. Click **OK**.

   The new details appear in the Calc Element Name dialog box.

To delete a Calc Element Name:

1. Click the Calc Element Name tab for the record on the data map.
2. Select the calc element entry.
3. Click the **Delete** icon.

Adding a Table to an IDMS Data Map

PowerExchange develops a relational view of the IDMS network environment. Additional tables might need to be created for particular applications.

To add a table to an IDMS data map:

1. Open the data map, and then right-click the map name.
2. Select **Add Table**. The Add Table dialog box appears.
3. Right-click on a record in the **Available Records** list, and select **Add Record**.

   Note: Capture data is only generated for the lowest level IDMS record within a table. When a table that contains information from two IDMS records is registered for capture, only data changes to the lowest level record within the table appear.

Adding Expressions

Additional information can be returned on each record by adding expressions to the data map. Expressions such as **GetDbKeyOfOwner** and **GetDataFlowType** can be carried through to logged capture information. This functionality is designed to help the creation of relational structures, possibly data warehouses, from IDMS source data.

For more information about using these expressions, see "Expressions Tab" on page 153.

Creating IMS Data Maps

PowerExchange requires IMS data maps to create SQL statements for relational-type access to IMS data during extraction processing. To use an IMS database as a data source, you must create a data map.

A data map maps IMS segments to table views of the data. You can use data maps to align fields, convert dates, and filter and enhance source data to improve data accuracy and reduce data volume.

In a data map, you can define one or more segments:

- Define a single segment in a data map if the associated data file contains only one segment type. A single-segment data map can define multiple tables.
- Define multiple segments in a data map if the source database contains multiple segment types. The fewer columns that PowerExchange must retrieve results in faster file processing and data extraction.

Note: For IMS data sources, *record* refers to a root segment and its children.

Task Flow for Creating Data Maps

Use the PowerExchange Navigator to add data maps to PowerExchange. When you create a data map, use the nonrelational view of data to define the layout of actual physical data by adding segment and field definitions. Provide the relational source view of data by defining tables and columns based on the record and field definitions.

You can use data maps for Lookup transformations to look up related values and determine if rows exist in an IMS target. When you create a data map for IMS, you can also use IMS as a source or target database.

To create IMS data maps, perform the following tasks:

1. In the PowerExchange Navigator, add an IMS data map. Enter basic information such as the schema name, data map name, access method, and IMS database name. When you create IMS data maps, you can select either the DL/1 BATCH or IMS ODBA access method. Also, you associate a data map with a single data file, which defines the record types and the data in the data map.

   For more information, see "Adding an IMS Data Map" on page 50.

2. Import the IMS DBD source into a data map. If the DBD source is available for the IMS data, import the DBD source to define segments and the hierarchical sequence for the data map. During the import, the IMS segments and fields are automatically mapped to tables, creating the relational format of source data that PowerExchange requires.

   For more information, see "Importing the IMS DBD Source into an IMS Data Map" on page 51.

3. Import a COBOL copybook for each IMS segment into the data map to overlay the segment with its COPYLIB.

   For more information, see "Importing Copybooks for IMS Segments in a Data Map" on page 93.

4. Send the data map to PowerExchange. This action stores the data map in PowerExchange by sending it to the PowerExchange Listener on the local or remote node. This action converts the data map into a platform-independent file that can be accessed by PowerExchange extraction processing.

For more information, see “Storing an IMS Data Map” on page 55.

5. Test the data map by using a database row test. A database row test accesses actual source data and displays it in table format. You can use a database row test to ensure that data can be retrieved from the source database.

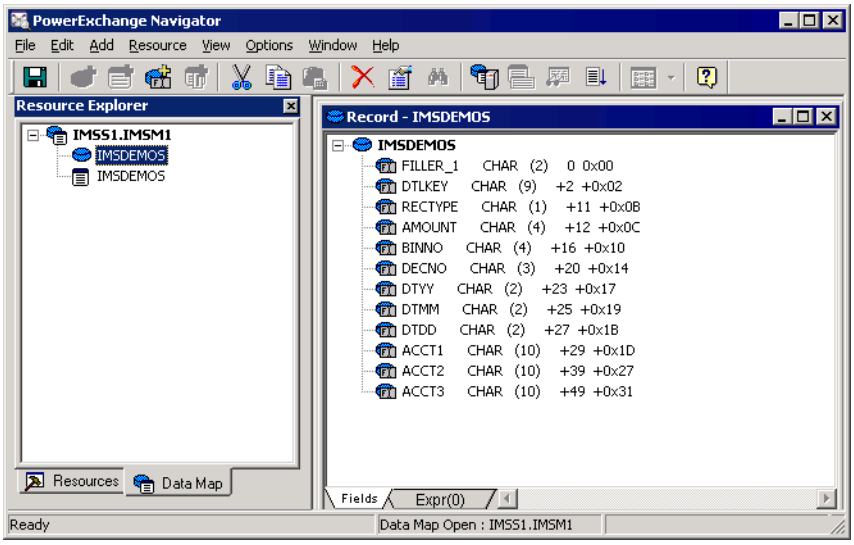
   For more information, see “Testing an IMS Data Map” on page 57.

Views of IMS Data

This section describes how to display the nonrelational, relational, and hierachical views of IMS data.

Nonrelational View

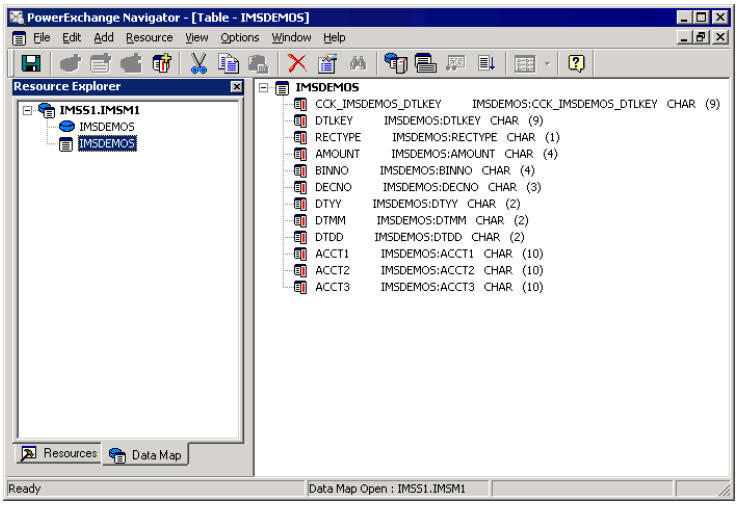
The following figure shows an IMS segment IMSDEMOS that was added to a data map during an import of the IMS DBD source:



When you import data for a single segment, the record and table view for the segment appears in the data map. If you import data for multiple segments, multiple segments appear in the data map.

Relational View

The following figure shows the IMSDEMOS record and table that was generated for an IMS segment during an import of the IMS DBD source:



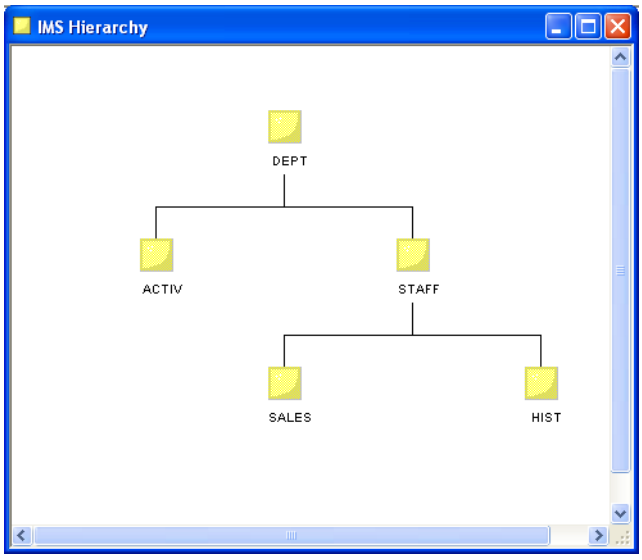
Hierarchical View

For each IMS data map that you create, the IMS hierarchy that results from the DBD source import is available.

To view an IMS hierarchy:

- On the Data Map tab, right-click a data map and click **Display IMS Hierarchy**.

  The IMS hierarchy is displayed, for example:



Example

The PowerExchange installation includes a basic IMSDEMOS example that provides a sample IMS DBD source that you can use to create a sample IMS data map. For more information, see "IMS Data Map Examples" on page 58.

Prerequisites for Creating a IMS Data Map

Before you create and test an IMS data map, gather the following IMS information:

- **IMS database to access**: IMS database and IMS database data set name.
- **PSB**: The PSB to use, and which PCB within the PSB to use.
  - **For IMS non-ODBA access.** Determine which PCB within a PSB to use. Count the number of PCBs in the PSB. If the PSB has COMPAT=YES, add one to that number.
  - **For IMS ODBA access.** Determine which PCB to use within a PSB. Note the PCB name because you must enter it when you create a data map.
  - **PROCOPT of the PCB.** To reduce the number of locking conflicts, make sure to set the PROCOPT to read-only for the PCB for the IMS database.
  - **PSB definition in the IMS SYSGEN.** If the PSB is used with the IMS ODBA interface or a Netport BMP job, make sure the PSB is defined in the IMS SYSGEN.
- **IMS subsystem ID**.
- **IMS segments**: Which segments to use, and if any segments are of variable length. Additionally, specifies the hierarchical sequence as defined in the DBD source.
- **IMS DBD source**: Data set location.
- **Update the PowerExchange configuration on MVS.** Update the DBMOVER member of the RUNLIB library. For more information, see "Configuring the PowerExchange DBMOVER Member on MVS" on page 49.

- **Update the PowerExchange configuration on Windows.** Update the dbmover.cfg file for the PowerExchange Listener on the Windows system where the PowerExchange Navigator runs. For more information, see "Configuring the PowerExchange DBMOVER File on Windows" on page 49.
- **Ensure that the data map represents a complete IMS segment.** If you plan to perform an INSERT or UPDATE to an IMS segment, the data map must represent the complete segment length as defined in the IMS DBD. Otherwise, an INSERT or UPDATE to the segment could result in nonblank data being written to the end of the segment that was not defined as FILLER.

  Note: To avoid this issue, you can add a FILLER definition to the COPYLIB before you import it to PowerExchange.

Configuring the PowerExchange DBMOVER Member on MVS

To access IMS data by using a data map, you must configure the DBMOVER member on MVS. The specific configuration steps depend on whether you access IMS by using DL/1 Batch or BMP, or IMS ODBA.

Note: You must restart the PowerExchange Listener to implement your configuration changes.

To use DL/1 Batch or BMP to access IMS:

1. Update the PowerExchange DBMOVER member in RUNLIB as follows:
   - Verify that the PSB name is correct in the PowerExchange LISTENER or NETPORT statement in the DBMOVER configuration member.
   - Add PowerExchange LISTENER or NETPORT entries if you use more than one PSB. For more information, see the *PowerExchange Reference Manual*.
2. Update the IMSJCL member in the PowerExchange RUNLIB library so that it works in your environment as either a DL/1 Batch or BMP job.

To use IMS ODBA to access IMS:

1. Include the RESLIB in the PowerExchange Listener STEPLIB, or verify that it is accessible through the MVS LNKLST concatenation.
2. Use a PSB that has PCBNAMES defined.

Configuring the PowerExchange DBMOVER File on Windows

Before you create a data map, update the dbmover.cfg file to provide access the IMS source data. The configuration file is located in the PowerExchange Navigator installation directory. Configuration settings depend on whether you access IMS by using DL/1 Batch or IMS ODBA.

Note: You must restart the PowerExchange Listener to implement your configuration changes. However, you do not need to start the PowerExchange Listener on Windows to access the IMS data.

To use DL/1 Batch or BMP to access IMS:

▶ Add a NODE statement that points to the Netport port for the PowerExchange Listener on MVS. The format is:

```
NODE=(location_name,TCPIP,[hostname_or_IPaddress],port_number)
```

To use IMS ODBA to access IMS:

▶ Add a NODE statement that points to the PowerExchange Listener port on MVS. The format is:

```
NODE=(location_name,TCPIP,[hostname_or_IPaddress],port_number)
```

Adding an IMS Data Map

When you add a data map in the PowerExchange Navigator, you must specify a schema name and a data map name and select an access method. The access method defines how PowerExchange connects to an IMS database to retrieve the source data. The data map appears in the Data Maps folder on the Resources tab as:

```
schema.datamapname
```

As the access method, you can select DL/1 Batch or IMS ODBA to define how to connect to the IMS database:

- The DL/1 Batch access method submits JCL to access the source data.

  Although the access method is DL/1 Batch access method, you can choose either DL/1 or BMP for the Netport job.
- The IMS ODBA method connects to an IMS database through an open source database connection.

Before you add a data map, complete the prerequisites listed in "Prerequisites for Creating a IMS Data Map" on page 48.

To add an IMS data map:

1. Click **Add > Data Map**.
2. In the Name dialog box, complete the following fields:

Field	Description
Schema Name	The name of the schema. Maximum length is 256 characters. Must begin with an alphabetic character.
Data Map Name	The name of the data map. Maximum length is 256 characters. Must begin with an alphabetic character.
Access Method	Specify one of the following access methods: - **DL/1 BATCH.** Accesses an IMS database through submitted JCL. - **IMS ODBA**. Accesses an IMS database through the PowerExchange Listener. - **SEQ**. Accesses an IMS database that you plan to use as a flat file.
Import Record Definitions	After the data map is created, the import copybook wizard opens.

3. Click **Next**.
4. Depending on the access method that you selected, perform one of the following actions:
   - If you selected the DL/1 Batch access method, continue to step 5.
   - If you selected IMS ODBA access method, go to step 6.
   - If you selected the SEQ access method, go to step 7.
5. In the DL/1 Batch Access Method dialog box, enter the following information and click **Next:**

Field	Description
IMS SSID	IMS subsystem ID, which can be up to four characters in length. Specify the value of the IMSID parameter in the DBMOVER configuration member. For more information about the IMSID parameter, see the *PowerExchange Reference Manual.* Specify this value only if you plan to read IMS unload data sets. **Note**: This IMS SSID value is not used in the DTL.DBWRITE.IMS resource profile that is used to control write access to databases accessed through DL/1 Batch. For more information about IMS security, see the *PowerExchange Reference Manual.*

Field	Description
DBD Name	Name of the IMS database description (DBD) for the IMS source. Specify this value only if you plan to read IMS unload data sets.
PCB Number	The relative PCB number for the database. If CMPAT=YES is specified in the PSB, add one to the PCB number. Specify this value only if you plan to read the IMS database by using a Net Port job that runs either DL/1 or BMP.
Data Codepage	Select the code page that describes the character set for the character data in the database. For more information, see “Data Map Code Pages” on page 72.
Skip First _ Records from File	If the file has one or more header records that are not required, specify the number of header records to skip.

6. In the IMS ODBA Access Method dialog box, enter the following information and click **Next**;

Field	Description
IMS SSID	IMS subsystem ID, which can be up to four characters in length. Specify the IMS ID for the control region. This field must match the value of the IMSID parameter specified in the MVS DBMOVER configuration member. For more information about the IMSID parameter, see the *PowerExchange Reference Manual*. **Note**: The RACF resource profile DTL.DBWRITE.IMS.*imsid* controls write access to databases accessed through ODBA. Verify that the *imsid* value matches the IMS SSID value for this field. For more information, see the *PowerExchange Reference Manual*.
DBD Name	Name of the IMS DBD for the IMS source.
PSB Name	Specify the name of a PSB that contains the specified DBD to be used to access this database. Enter up to eight alphanumeric characters. Do not include special characters.
PCB Name	For the specified PSB, enter the named PCB that references the specified DBD. You must either give the PCB a label or specify the PCD in the **PCB Name** box.
Data Codepage	Select the code page that describes the character set for the character data in the database. For more information, see “Data Map Code Pages” on page 72.
Skip First _ Records from File	If the file has one or more header records that are not required, enter the number of records to skip.

7. In the SEQ Access Method dialog box, enter or browse to the flat file name.
8. Click **Finish**.
9. If you selected the **Import Record Definitions** option, import the IMS DBD source to the data map. For more information, see “Importing the IMS DBD Source into an IMS Data Map” on page 51.

Importing the IMS DBD Source into an IMS Data Map

Use the Import Copybook options in the PowerExchange Navigator to access and import the IMS DBD source. You import the IMS DBD source into a data map to define the segments and the hierarchical sequence,

generate a hierarchical schema, and add search fields and CCK fields to segment definitions. This provides the relational view of data PowerExchange needs to run SQL-type extracts against the source data.

When you import a DBD source, the actual DBD source hierarchy is preserved in the IMS database. The DBD source representation is placed in the local PowerExchange library.

Before you import the IMS DBD source, you should know the name of the IMS data set where the DBD source member resides.

PowerExchange creates a relational view based on the import. The relational view maps a table to each IMS segment and columns to each IMS field.

After you import the IMS DBD source, import a copybook for each segment. For more information, see "Importing Copybooks for IMS Segments in a Data Map" on page 93.

When the data map is complete, send it to a PowerExchange Listener on the local or remote node. This converts the data map to a platform-independent file. You can also test a data map by using a database row test. A database row test accesses actual source data and displays it in table format. You can use a database row test to ensure that data can be retrieved from the source database. For more information, see "Testing an IMS Data Map" on page 57.

Before you import a DBD source into a data map, complete the prerequisites for creating a data map. For more information, see "Prerequisites for Creating a IMS Data Map" on page 48.

To import the IMS DBD source:

1. If you selected the **Import Record Definitions** option, the Import Copybook - Source Details dialog box appears when you complete the data map. If the dialog box does not appear, select the data map on the Resources tab and click **File > Import Copybook**.
2. Enter the following information:

Field	Description
Source	Select one of the following options: - **Local**. Indicates that the DBD is local. The data has been downloaded and stored locally. - **Remote**. The data is downloaded from a remote location, such as MVS.
Type	As the type of import to perform, select **DBD**.
Column Range	Define the start and end columns. Typically, for a DBD source, this range is 1 to 72.
Last Import	Click **Last Import** to display import copybook information. The current import copybook specifications are those of the previous import.

3. If you selected **Local** as the **Source** option, enter the following information in the Import Copybook - Local DBD Details dialog box:

Option	Description
File Name	Name of the DBD to import.
Preview	Click **Preview** to preview the DBD.

4. If you selected **Remote** as the **Source** option, enter the following information in the Import Copybook - Remote DBD Details dialog box and click **Next:**

Field	Description
File Name	PDS file name and member in the following format: `HLQ.DTLDEMO(dbdname)`
Location	Remote NODE in the DBMOVER configuration member.

Field	Description
UserID	MVS user ID if security has been implemented.
Password	MVS password if security has been implemented.
Save File Locally As	File name under which to save the copybook when it is retrieved from the remote platform.
Preview	Click **Preview** to preview the DBD source.

5. In the Import Copybook - Configuration Details dialog box, enter the following information:

Field	Description
Prompt on record import	When a record is created, prompt for a record name showing the first name found.
Prompt on field import	When a field is created, prompt for a name showing the name found.
Prompt on table creation	When a table is created, prompt for a table showing the default name.
Create table on each record imported	Creates a table if a record is imported.
Create tables for DL1 hierarchical paths	Select when you work with DBDs to have all tables created for all DL/1 hierarchical paths. Only available for a DBD source.
Select first data redefinition	Select so that if a REDEFINES clause is found, the first data definition is automatically used.
Start import automatically	Select to automatically start the import.
Action on duplicate record	Select the action to take if a record with a duplicate is found: - PROMPT - Provide a UNIQUE NAME - OVERWRITE it - SKIP it - APPEND
Action on duplicate field	Select the action to take if a field with a duplicate is found: - PROMPT - Provide a UNIQUE NAME - OVERWRITE it - SKIP it
Action on duplicate table	Select the action to take if a table with the same name is found: - PROMPT - Provide a UNIQUE NAME - OVERWRITE it - SKIP it

6. Click **Finish**.
7. Review your selections in the Import Copybook Information window.
8. If the options are correct, click **OK**.
9. Perform one of the following actions:
   - If you chose to be prompted during the import, continue to “Using the Import Prompts” on page 94.
   - If you did not choose to be prompted, review the data that is imported.
10. After importing a DBD source into a data map, import a COBOL copybook for each segment that was imported. For more information, see “Importing Copybooks for IMS Segments in a Data Map” on page 93.

Returning an IMS RBA for a User-Defined Field

You can have PowerExchange return an RBA to create a user-defined field in an IMS data map. Use the Expressions options.

To return an RBA:

1. In the Resource Explorer, double-click the data map and then select the record on the Data Map tab that represents the IMS segment for which you want to create a user-defined field.
2. Click the Expr tab.

   Note: If no expression exists, a zero appears in parenthesis.
3. Right-click within the Expr tab and click **Add Field at End**.

   In the Expressions dialog box, enter the following information:

Option	Description
Name	Name to describe the RBA.
Type	Data type of the field.
Prec	Number of digits in numeric data, if applicable.
Scale	Number of digits to the right of the decimal point, if applicable.
Length	Length of the field.
Phase	Select one of the following options: - R (Read) - W (Write) - RW (Read and Write)
Expression	Expression used to populate the field.
Validation	Displays a message if a problem with the field occurs.

Modifying an IMS Data Map

The PowerExchange Navigator provides options for modifying a data map by selecting segments and tables and manually changing the properties. Instead of manually changing properties, re-import the IMS DBD source or COBOL copybook if you need to update segments and tables.

Warning: Modifying segments and tables manually can result in loss of data in a data map.

For more information, see “Importing the IMS DBD Source into an IMS Data Map” on page 51 and “Importing Copybooks for IMS Segments in a Data Map” on page 93.

Storing an IMS Data Map

Store each data map that you create so that PowerExchange can extract the related source data. Storing a data map involves sending the data map to the PowerExchange Listener on the same node as your IMS source data.

This process converts the data map to a platform-independent file that can be accessed by PowerExchange extraction processing.

To send an IMS data map to the PowerExchange Listener:

1. In the Resource Explorer, select a data map and click **File** > **Send to Remote Node**.
2. In the Data Map Remote Node dialog box, enter the following information:

Option	Description
User ID	Your user ID if security is set up on the remote MVS node.
Password	Your password if security is set up on the remote MVS node.
Location	Location (NODE from the dbmover.cfg file) to send the data map to. This platform is the one from which the data that is extracted resides.
File Password	File password if the file is password protected.
Save User ID and Password(s) for session.	Saves the user ID and password you entered for the rest of the session.

3. Click **OK**.

   After you store the data map, you can test the data source. For more information, see "Testing an IMS Data Map" on page 57.

Using Lookup Transformations with IMS Databases

You can use lookup transformations in PowerCenter to lookup data in IMS databases. You should use a separate Netport job for each lookup transformation for PCBs with a "GO*x*" PROCOPT.

Note: Specify the IMS key values of a segment so that SSAs perform efficiently within IMS.

For more information about using a lookup transformation for IMS databases, see *PowerExchange Interfaces for PowerCenter*.

Writing Data to IMS Databases

You can write source data from a data map to an IMS database. When you write data to IMS, specify the key values of a segment so that all the processing involved in writing data for the segment takes place within IMS.

Use PowerExchange Client for PowerCenter (PWXPC) to write data to IMS. For more information about PWXPC, see *PowerExchange Interfaces for PowerCenter*.

When you write data to IMS, use the following guidelines:

- You cannot use SDEP segments.
- You cannot update a segment below an unkeyed segment.
- You can use segments that contain an OCCURS clause where the COPYLIB has been changed to identify each of the fields in the OCCURS clause as a separate field, which eliminates the OCCURS clause in the COPYLIB definition. You cannot generate a separate row for each occurrence of the OCCURS clause.
- When you import a data map to PowerCenter, make sure any CCK fields that you want to use as key fields are marked as key fields.
- In PowerCenter, use a lower Commit frequency to reduce the risk of locking segments.

- A separate NETPORT job is recommended for writing IMS data. This allows you to access a PSB with write intent and modify the JCL to support updating IMS data, such as updating the IEFRDER log.
- When you write data for one of the following types of fields, indicate the field is optional as you create a data map:
  - When you plan to move spaces for a field to the database
  - When you plan to move no data for a specific field to the database

  If you have several fields in one of these situations, you only need to specify that the first field in the data map is optional, and all fields following that field are considered optional. If you do not specify the field is optional, errors appear in the PowerCenter session log.

To write data to IMS databases:

1. Ping the IP address of the MVS system where the target database is located to make sure it is accessible.
2. Add a NODE statement that points to the remote platform in the PowerExchange configuration file dbmover.cfg.
3. Use the following command to test that the remote PowerExchange Listener is started:

   ```
   dtlrexe loc=xxxx prog=ping uid=<userid> pwd=<pwd>
   ```

   Where *xxxx* is the NODE name from the PowerExchange configuration file dbmover.cfg.
4. Use PowerExchange to create a data map. For information about creating an IMS data map, see "Creating IMS Data Maps" on page 46. To create data maps for other databases, see the appropriate PowerExchange CDC guide.
5. Store the data map.
6. Test the data map. For more information, see "Testing an IMS Data Map" on page 57.
7. Use PowerCenter to execute the request to move the data to IMS. For more information, see *PowerExchange Interfaces for PowerCenter*.

View or Change IMS Options

Use the IMS Options for a data map that is used to write to an IMS database. These options provide default values that are set for optimal performance when working with IMS.

Changing the default settings impacts the amount of data sent to the database and the time required to write the data. Because of these options, PowerExchange regards IMS as if it is a relational target. The defaults are set so that PowerExchange acts like a typical IMS COBOL application program.

Warning: Do not change the Delete Options default settings unless you are familiar with the IMS hierarchy. Changing these settings can result in loss of IMS data.

To view or change IMS options:

1. On the Data Map tab, right-click a table and select **Properties**.
2. Click the IMS Options tab and complete the following options:

Option	Description
Update/Insert Options	Indicates which segment level to apply the update or insert to. Select one of the following options: - Lowest segment only. Applies the action to the lowest segment only. - All possible levels. Applies the action to all segment levels.
Update all matching segments	Indicates all non-unique matching segments should be updated. **Note:** You cannot select this option for complex tables.

Option	Description
Delete Options	Indicates which segment level to apply the delete. Select one of the following options: - Lowest segment only. Applies the delete action to the lowest segment only. - All childless segments in hierarchy. Applies the delete action to all childless segment levels. **Warning**: Do not change the Delete Options default settings unless you are familiar with the IMS hierarchy. Changing these settings can result in loss of IMS data.
Delete all matching segments	Indicates all non-unique matching segments should be deleted.
Ignore “Record not found” error on delete	Indicates all “Record not found” errors that are generated by the delete action should be deleted.

Testing an IMS Data Map

After you create a data map and store the data map in PowerExchange, you can test the data source by using a database row test. A database row test accesses actual source data and displays it in table format.

You can use a database row test to ensure that data can be retrieved from the source file or database. A database row test can read data from an IMS unload file or an IMS database.

The following table describes the fields on the Database Row Test dialog box:

Field	Description
DB_Type	Select one of the following DB types: - **IMSUNLD**. Data is read from an IMS unload file. - **NRDB** or **NRDB2**. Data is read from an IMS database. **Note**: Select the NRDB or NRDB2 DB type for nonrelational databases. Use NRDB2 for data maps with two-tier naming.
Location	The location of the PowerExchange Listener, as specified in the DBMOVER configuration file. This location must be a Netport location. The port number must correlate to the location in the DBMOVER configuration file on MVS. The NETPORT statement points to a RUNLIB member called IMSJCL when the product is installed. By using this location, the IMSJCL job is submitted on MVS.
Override File Name	Not applicable to IMS.
File Password	Not applicable to IMS.
UserID	User ID if security is activated for the PowerExchange Listener.
Password	Password if security is activated for the PowerExchange Listener.
Fetch	Select one of the following options: - **Columns**. Retrieves information about columns. - **Data**. Retrieves actual data rows. - **Foreign keys**. Retrieves information about Foreign keys. - **Primary keys**. Retrieves information about Primary keys. - **Procedure Cols**. Retrieves information about particular stored procedures. - **Procedures**. Retrieves information about available stored procedures. - **Records**. Retrieves information about records. - **Schemas**. Retrieves information about schemas. - **Tables**. Retrieves information about tables.
SQL Statement	Generated select statement in the format: `select * from schema.map_tablename(segment)`

The following procedures identify which options to use for each type of IMS data source.

To test an IMS data map sourced from an IMS database:

1. In the Resource Explorer, double-click the data map.
2. On the Data Map tab, select a table. Then click **File > Database Row Test**.
3. In the message box that prompts you to send the data map to a remote location, click **Yes**.
4. In the Data Map Remote Node dialog box, select the remote node from which the IMS data is read.
5. Click **OK**.
6. In the Database Row Test dialog box, perform the following actions:
   - In the **DB Type** list, click **NRDB** or **NRDB2**.
   - In the **Fetch** list, click **Data**.
7. Click **Go**.

   The Database Row Test Output window displays the output.
8. Close the Database Row Test Output window.
9. Close the Database Row Test dialog box.
10. To save the data map, click **File > Save**.
11. Click **File > Close Resource**.

    The Data Map Remote Node dialog box is displayed.

To test an IMS data map sourced from an IMS unload file:

1. In the Resource Explorer, double-click the data map.
2. On the Data Map tab, select a table. Then click **File > Database Row Test**.
3. In the message box that prompts you to send the data map to a remote location, click **Yes**.
4. In the Data Map Remote Node dialog box, select the remote node from which the IMS data is read.
5. Click **OK**.
6. In the Database Row Test dialog box, perform the following actions:
   - In the **DB Type** list, click **IMSUNLD**.
   - In the **File Name** box, enter the name of the IMS unload file.
   - In the **Fetch** list, click **Data**.
7. Click **Go**.

   The Database Row Test Output window displays the output.
8. Close the Database Row Test Output window.
9. Close the Database Row Test dialog box.
10. To save the data map, click **File > Save**.
11. Click **File > Close Resource**.

IMS Data Map Examples

These examples provide a sample IMS DBD source and steps to create a single-segment data map and a multiple-segment data map using the sample DBD source.

Before you create the example data maps, review “Prerequisites for Creating a IMS Data Map” on page 48.

IMS DBD Source

The following IMS DBD source example is provided with your PowerExchange installation:

```
DBD    NAME=IMSDEMO,ACCESS=(HDAM,VSAM),X
       RMNAME=(DFSHDC40,50,80),X
       EXIT=(*,KEY,PATH,(CASCADE,KEY,PATH),LOG)
DATASET DD1=IMSDEMO,DEVICE=3380,SIZE=8192,SCAN=3
SEGM NAME=IMSDEMOS,BYTES=80FIELD NAME=(DTLKEY,SEQ),BYTES=9,START=3
FIELD NAME=RECTYPE,BYTES=1,START=12FIELD NAME=AMOUNT,BYTES=4,START=13
FIELD NAME=BINNO,BYTES=4,START=17FIELD NAME=DECNO,BYTES=3,START=21
FIELD NAME=DTYY,BYTES=2,START=24FIELD NAME=DTMM,BYTES=2,START=26
FIELD NAME=DTDD,BYTES=2,START=28FIELD NAME=ACCT1,BYTES=10,START=30
FIELD NAME=ACCT2,BYTES=10,START=40
FIELD NAME=ACCT3,BYTES=10,START=50
DBDGEN
FINISH
```

When you install PowerExchange on MVS, a number of libraries are created with the high-level qualifier that you specified in the MVS Installation Assistant. On MVS, run the following job to create the IMS environment for the example installation:

```
hlq.DTLDEMO(IMSDEF)
```

Example 1. Creating an IMS Single-segment Data Map

This example creates a data map for a single IMS segment.

In this example, you use the IMSDEMOS example and import a DBD source to define the segment and the hierarchical sequence. You also import a COBOL copybook for the segment. This example uses the DL/1 BATCH access method.

To create a single-segment IMS data map:

1. Click **Add > Data Map**.
2. In the Name dialog box, perform the following actions:
   - In the **Schema Name** box, enter **IMSS1**.
   - In the **Data Map Name** box, enter **IMSM1**.
   - In the **Access Method** list, click **DL/1 BATCH**.
   - Select the **Import Record Definitions** option.
3. Click **Next**.
4. In the DL/1 Batch Access Method dialog box, click **Finish**.

   Note: The **IMS SSID** and **DBD Name** values are required only if you are reading an IMS unload data set. The **PCB Number** value is required if you are reading the IMS database by using a Netport job that runs either DL/1 or BMP.

   On the Data Map tab, the IMSS1.IMSM1 data map appears.
5. In the Import Copybook - Source Details dialog box, perform the following actions:
   - Select the **Remote** option.
   - In the **Type** list, select **DBD**.
6. Click **Next**.

   Note: The following steps demonstrate how to import a DBD to define segments and the hierarchical relationships, with data in columns 1 through 80.

7. In the Import Copybook - Remote DBD Details dialog box, perform the following actions:
   - In the **File Name** box, enter the PDS name and DBD member with the high-level qualifier (HLQ) that you specified at installation:

     hlq.DTLDEMO(IMSDBD)
   - In the **Location** list, select your MVS location. If this connection is the first MVS connection you are using for PowerExchange, add a NODE statement to the dbmover.cfg configuration file on Windows where the PowerExchange Navigator is installed.
   - If security has been implemented for the PowerExchange Listener on MVS, enter the MVS user ID and password.
   - In the **Save File Locally As** box, enter **IMSDBD**.
8. Click **Next**.
9. In the Import Copybook - Configuration Details dialog box, ensure that the **Create tables for DL1 hierarchical paths** option is selected. Accept the other defaults.
10. Click **Finish**.
11. In the Import Copybook Configuration Details dialog box, review your selections.
12. Click **Finish**.
13. Click **OK** to import the DBD source.
14. In the Record Definition dialog box, click **Import**. If necessary, skip a record type or stop an import.
15. To complete the import, use one of the following the tools:

Tool	Description
Next Redefinition icon **Previous Redefinition** icon	Moves the blue arrow on the left to point at the line that you want to select.
Resume Import icon	Resume the import if prompted.

16. View the log information for the import process.

To import the copybook:

1. In the Resource Explorer, double-click the data map.
2. On the Data Map tab, select the IMS segment. Click **File > Import Copybook**.
3. On the Import Copybook - Source Details dialog box, perform the following actions:
   - Select the **Remote** option.
   - In the **Type** list, select **COBOL**.
4. Click **Next**.
5. In the Import Copybook Remote DBD Details dialog box, perform the following actions:
   - In the **File Name** box, enter the copybook name and location.
   - In the **Location** list, select your MVS location.
   - If security has been implemented for the PowerExchange Listener on MVS, enter the MVS user ID and password.
   - In the **Save File Locally As** box, enter a file name.
6. Click **Next**.
7. In the Import Copybook - Configuration Details dialog box, accept the defaults.
8. Click **Finish**.
9. In the Import Copybook Configuration Details dialog box, review your selections.

10. Click **Finish**.
11. Respond to the import prompts. For more information, see "Using the Import Prompts" on page 94.
12. When the import is complete, review the data.
13. You can store the data map with PowerExchange and test the data map. For more information, see "Testing an IMS Data Map" on page 57.

Example 2. Creating an IMS Multiple-segment Data Map

This example defines a complex table. You define a complex table by adding multiple segments, one at a time, in the **Record Dependencies** list to create a hierarchical relationship between these segments.

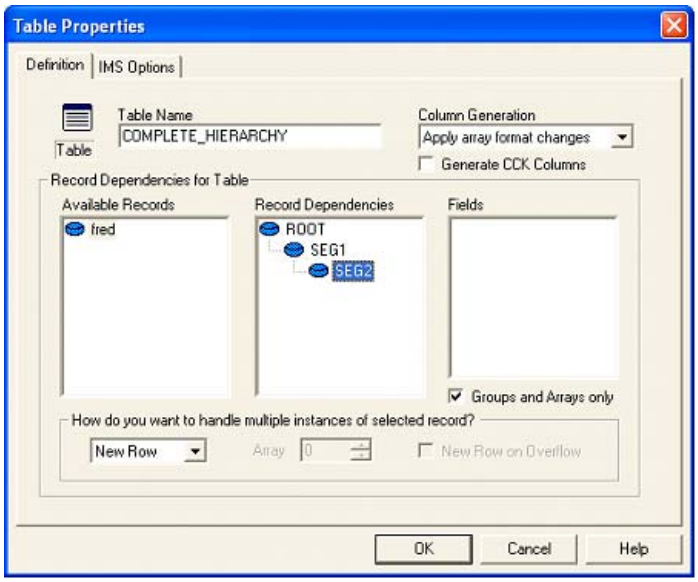
To define a hierarchical relationship among segments, you move segments from the **Available Records** list to the **Record Dependencies** list.

The following procedure describes how to define a hierarchy. For information about viewing an IMS hierarchy, see "Hierarchical View" on page 48.

To create a hierarchy:

1. On the Resources tab, double-click a data map that contains multiple segments.
1. On the Data Map tab, right-click a table and click **Properties**.
2. In the Table Properties dialog box, perform the following actions:
   - In the **Record Dependencies** list, click a segment to establish it as the parent.
   - In the **Available Records** list, right-click a segment that you want to add as a child and click **Add Record as Child**.
   - In the **Record Dependencies** list, click the parent segment.
   - In the **How do you want to handle multiple instances of selected record?** list, click **New Row**. For the multiple instances option, you can also select either **Ignore** or **Array**. For more information about these options, see "Complex Table Options" on page 16.
   - In the **Record Dependencies** list, click each child segment and set the multiple instances option to **New Row**.
3. Click **OK**.

   The following figure shows a hierarchy for multiple segments:



Note: You can set a different options for **How do you want to handle multiple instances of selected record?** for each segment in the hierarchy. For example, you could set ROOT to **Ignore**, SEG1 to **New Row**, and SEG2 to **Array**.

If a parent segment is set to **Array**, all child segments must be set to **Ignore**.

You can also store segments at the same level. For example, you could add the SEG1 and SEG2 to ROOT segments as siblings.

For more information about defining complex tables, see "Defining Complex Tables" on page 15.

Creating VSAM Data Maps

PowerExchange enables access to nonrelational data through the use of SQL, as though the data source was relational.

To access VSAM data through PowerExchange, you must create a data map using the PowerExchange Navigator. Use the PowerExchange Navigator to build data maps and personal metadata profiles for VSAM data sets.

PowerExchange can access the following types of VSAM data set:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)
- Fixed-length relative record data set (RRDS)
- Variable-length relative record data set (VRRDS)

These types of VSAM data sets can be basic format, extended-format, or compressed data sets.

To create a data map, you must start the PowerExchange Navigator, either from the shortcut created when PowerExchange was installed or by running dtlui.exe from the Windows **Start > Run** option.

To create a VSAM data map:

1. Click **Add > Data Map**.
2. In the Name dialog box, enter a schema name and a data map name, and select the appropriate access method for the type of VSAM data set.

   Note: Select **RRDS** for both fixed-length and variable length RRDS (VRRDS) data sets.
3. Select the **Import Record Definitions** box to import the record layout for the VSAM data set.
4. Click Next.

   The Access Method dialog box for the VSAM data set type displays.

   The ESDS Access Method dialog box displays the following fields:

Field	Description
File Name	The fully-qualified data set name of the VSAM data set.
Number of Data Buffers	The number of I/O buffers for data control intervals that PowerExchange requests VSAM to allocate. The default is 2.

Field	Description
CI ACCESS	If selected, specifies that PowerExchange should access the entire contents of a control interval rather than individual data records when reading VSAM data sets. This option improves read performance. For more information, see “Improving Bulk Read Performance for VSAM Data Sets” on page 65. **Note:** CI ACCESS cannot be used if the data set is compressed.
Prefix record with XRBA value	If selected, specifies that PowerExchange should return the extended relative byte address (XRBA) value for all records read from the VSAM data set. **Note:** You must has include a 8-byte binary field at the beginning of the record to contain the XRBA value.
File List Processing	If selected, specifies that PowerExchange should us file list processing for this data map. For more information about file list processing, see “File List Processing” on page 103.
Data Codepage	The code page that describes the character set for the character data in the database. For more information, see “Data Map Code Pages” on page 72.
Skip First Records from File	PowerExchange skips the specified number of records before returning data. The default is 0.

The KSDS Access Method dialog box displays the following fields:

Field	Description
File Name	The fully-qualified data set name of the VSAM data set.
Number of Data Buffers	The number of I/O buffers for data control intervals that PowerExchange requests VSAM to allocate. The default is 2.
Number of Index Buffers	The number of I/O buffers for index control intervals that PowerExchange requests VSAM to allocate. The default is 1.
CI ACCESS	If selected, specifies that PowerExchange should access the entire contents of a control interval rather than individual data records when reading VSAM data sets. This option improves read performance. For more information, see “Improving Bulk Read Performance for VSAM Data Sets” on page 65. **Note:** CI ACCESS cannot be used if the data set is compressed.
Data Codepage	The code page that describes the character set for the character data in the database. For more information, see “Data Map Code Pages” on page 72.
Skip First Records from File	PowerExchange skips the specified number of records before returning data. The default is 0.

The RRDS Access Method dialog box displays the following fields:

Field	Description
File Name	The fully-qualified data set name of the VSAM data set.
Number of Index Buffers	The number of I/O buffers for index control intervals that PowerExchange requests VSAM to allocate. The default is 1.
Number of Data Buffers	The number of I/O buffers for data control intervals that PowerExchange requests VSAM to allocate. The default is 2.
CI ACCESS	If selected, specifies that PowerExchange should access the entire contents of a control interval rather than individual data records when reading VSAM data sets. This option improves read performance. For more information, see “Improving Bulk Read Performance for VSAM Data Sets” on page 65. **Note:** CI ACCESS cannot be used if the data set is compressed.
Prefix record with RRN value	If selected, specifies that PowerExchange should return the relative record number (RRN) value for all records read from the VSAM data set. **Note:** You must has include a 8-byte binary field at the beginning of the record to contain the RRN value.
Data Codepage	The code page that describes the character set for the character data in the database. For more information, see “Data Map Code Pages” on page 72.
Skip First Records from File	PowerExchange skips the specified number of records before returning data. The default is 0.

In the **File Name** field, enter the data set name for the VSAM data set.

By default, PowerExchange uses the system-defined buffering for VSAM processing. To tune PowerExchange processing, specify additional VSAM buffers. For more information about buffer selection, see “Improving Bulk Read Performance for VSAM Data Sets” on page 65.

5. Click **Finish**.

The Import Copybook - Source Details dialog box displays. This dialog box has the following fields:

Field	Description
Source	The location of the copybook or metadata information. Select **Local** if the metadata is available on this Windows machine or **Remote** if the metadata is remote from this Windows machine.
Source Type	The format of the copybook, which is one of the following types: - **COBOL.** COBOL copybook - **PL/1.** PL/I copybook The default is COBOL.
Column Range	The start and end columns that should be inspected. Typically this should be set to 7-72 for COBOL copybooks and 1-72 for PL/1 copybooks. Available range is 1 through 999.

6. Click **Next**.
7. In the Import Copybook - Remote Details dialog box, enter the following information:

Setting	Description
File Name	The fully-qualified MVS data set name, including member name in parentheses if a PDS, that contains the copybook source.
Location	The node name of the MVS system as defined in the dbmover.cfg file.
UserId/Password	If PowerExchange is configured for security, a valid MVS user ID and password.
Save File Locally As:	File into which the copybook is saved locally.
Preview	Click this button to preview the copybook source.

8. Click **Next**.
9. In the Import Copybook - Configuration Details dialog box, select options.
10. Click **Finish**.
11. The Record Definition dialog box displays with the VSAM record name in the **Name** field.
12. Click **OK**.

    The data map definition is now complete.
13. Close and save the new data map.

Improving Bulk Read Performance for VSAM Data Sets

PowerExchange provides the following options to improve the performance of bulk read operations for VSAM data sets:

- Specify the number of data and index buffers to use when reading data.
- Specify control interval access.
- Use keys in WHERE clauses.

Number of Data Buffers

You can specify the number of I/O buffers VSAM is to use for transmitting data between virtual and auxiliary storage. A buffer is the size of a control interval in the data component. Number of Data Buffers (BUFND) must be a number between 0 and 65535. The minimum number you can specify is 1 plus the number specified for STRNO. If you omit STRNO, BUFND must be at least 2, because the default for STRNO is 1.

Note: The minimum buffer specification does not provide optimum sequential processing performance. Generally, the more data buffers that are specified, the better the performance. Additional data buffers can benefit direct inserts or updates during control area splits and benefits spanned record accessing.

The maximum number of buffers allowed is 255: 254 data buffers and 1 insert buffer.

Number of Index Buffers

Specifies the number of I/O buffers VSAM is to use for transmitting the contents of index entries between virtual and auxiliary storage for keyed access. A buffer is the size of a control interval in the index. Number of Index Buffers (BUFNI) must be a number between 0 and 65535. The minimum number is the number specified for STRNO: If you omit STRNO, BUFNI must be at least 1, because the default for STRNO is 1. The default is the minimum number required.

Additional index buffers improve performance by providing for the residency of some, or all, of the high-level index, thereby minimizing the number of high-level index records retrieved from DASD for key-direct processing.

The default is the minimum number required. The maximum number of buffers allowed is currently 255 (254 data buffers and 1 insert buffer).

The number of buffers used can be altered at a PowerExchange system level by entering a VSAM parameter line in the DBMOVER configuration PDS member which is documented in the *PowerExchange Reference Manual*. If this parameter is present, it overrides any entry in the data map.

Use Control Interval Access

Control interval (CI) access is a read option that causes the VSAM access method to use direct control interval reads to retrieve data records. If you extract all of the data from the VSAM data set, this option may improves the sequential read performance. If you specify a WHERE clause when retrieving the data, CI access provides no performance benefit.

To select at the data map level, select the CI ACCESS option on the Access Method tab of the data map. The **CI ACCESS** option is valid for KSDS, ESDS, and RRDS access methods.

CI access is not supported for compressed VSAM data sets or VSAM data sets with spanned records.

Use VSAM Keys

PowerExchange reads the key information for VSAM data sets from the catalog when the data set is allocated and opened. Both the key length and the relative key position are retrieved.

For a record layout (keypos=10, keylen=4) similar to

```
01 REC.
    04 HEADER     PIC X(10).
    04 KEY.
        08  KEY1     PIC 99.
        08  KEY2     PIC 99.
    04 REST          PIC X(200).
```

Note:

- KEY1 must be in the WHERE clause for any optimization to work.
- KEY2 is used if KEY1 is also in the WHERE clause.

The key information is used to speed up the processing of WHERE clauses, which provides optimization. Different formats of the WHERE clause produce different results. For example:

```
Line 1    SELECT * FROM TEST.VSAM1.TAB WHERE (KEY1 > 50) AND (KEY1 < 90)
```

This example generates the same results as the following example, but with a different elapsed time:

```
Line 2    SELECT * FROM TEST.VSAM1.TAB WHERE (KEY1 > 50  AND  KEY1 < 90)
```

Line 1 gets translated into (50 < KEY1 < HIGH-HALUES) AND (LOW-VALUES < KEY1 < 90) When these two are combined, the result is (KEY1 > LOW-VALUES AND KEY1 < HIGH-HALUES)- the entire file. This format is still technically optimized, but when the file is processed the read start is positioned at LOW-VALUES and read until HIGH-VALUES - not ideal.

Line 2 is the preferred option. The line 2 syntax positions the file at record 50 and read from there until KEY1 >= 90.

Optimization is used for key positioning only, the data is selected against the WHERE clause to ensure the correct records are filtered.

Note: If skip records > 0 is specified, using the Skip First 'n' Records from File field on the KSDS Access Method Wizard, the SELECT statement is not optimized even if KEY1 is specified.

Retrieving the RRN or RBA for Records in VSAM Data Sets

PowerExchange can return the Relative Record Number (RRN) or Relative Byte Address (RBA) for ESDS and RRDS data sets in the following ways:

- Prefix records with the RRN or XRBA value when you select these options in the data map properties
- Include the RRN or RBA value in a new field in the record when you use the GetDatabasekey expression

Note: You cannot use the GetDatabasekey expression if you request off load processing.

Retrieve the RRN or RBA by Using Data Map Options

You can use options in the data map to retrieve the relative record number (RRN) or relative byte address (RBA) for VSAM data sets.

To retrieve the RRN or RBA:

1. Open the data map.
2. Right-click on the data map name, and then select **Properties**.
3. The Data Map Properties dialog box displays. Click the Access Method tab.
4. On the Access Method tab, select **Prefix record with XRBA** to retrieve the RBA for ESDS data sets or **Prefix Record with RRN** to retrieve the RRN for RRDS data sets.
5. If you request that PowerExchange return the RRN or RBA value, you must add a field at the beginning of the record definition to hold this value.

   Click the record in the data map to open the record layout dialog box.
6. Right-click on the first field in the record, and then select **Add Field Before**.
7. Enter a value for **Field Name**. Select **BIN** for the **Field Type** and **8** for the **Length** field.
8. Click **OK**.
9. Click on the table to open the table layout dialog box.
10. Right-click on the first column in the table, and then select **Add Column Before**.
11. Enter a name for the new column in the **Name** field, and then select the new field from the list in **Base Field**.
12. Click **File > Send to Remote Node** to send the altered data map to the MVS system where the VSAM data set resides.
13. Close the data map.

Retrieve the RRN or RBA by Using the GetDatabaseKey Expression

You can use the GetDatabaseKey expression to retrieve the relative record number (RRN) or relative byte address (RBA) for VSAM data sets. You cannot use the GetDatabaseKey expression if you use off load processing.

You specify expressions in the record definition of the data map and create user-defined fields to hold the results of the expression.

To retrieve the RRN or RBA using expressions:

1. Open the data map and click on the record to open it.
2. On the Expr tab, right-click in the window and click **Add Field at End**.
3. In the **Name** column, enter a name.
4. In the **Type** column, enter **BIN**.
5. In the **Length** column, enter **8**.

6. In the **Expression** column, enter **GetDatabasekey**().

   If the expression field is valid, the PowerExchange Navigator displays a green check mark. Otherwise, the PowerExchange Navigator displays a red cross and a message.

7. After the field has been added to the record, you must include it in the table representation. Click the table to open it.
8. Right-click the table name and click **Add Column at End**.
9. In the list in **Base Field**, select the new column.
10. In the **Name** field, enter a name for the column.
11. Click **OK**.
12. Click **File > Send to Remote Node** to send the altered data map to the MVS system where the VSAM data set resides.
13. Close the data map.

Testing a VSAM Data Map

You can test a data map by using a database row test. A database row test accesses actual source data and displays it in table format. You can use a database row test to ensure that data can be retrieved from the source database.

For more information, see "Using Database Row Test" on page 141.

Creating C-ISAM Data Maps

To access C-ISAM data using PowerExchange, the structure of the file, tables, and columns must be known, so that you can define a data map.

Step 1. Add a C-ISAM Data Map

Use the following procedure to add a C-ISAM data map.

To add a C-ISAM data map:

1. In the Name dialog box, perform the following actions:
   - Enter a **Schema Name**, such as **ISAM**, and a **Data Map Name**, such as map1.
   - For C-ISAM access, select the ISAM access method.
   - Clear the **Import Record Definitions** option. You specify record definitions later.
2. Click **Next**.
3. On the ISAM Access Method dialog box, enter the full path and file name of the C-ISAM data file.
4. Click **Finish**.

Step 2. Add a Record

Use the following procedure to add a record.

To add a record:

1. Right-click the data map and click **Add Record**.

   The Add Record dialog box appears. The name is for reference when building tables.

2. Enter the required record name, such as record.

3. Click **OK**.

   The Record window appears.

Step 3. Add a Field

Three fields in the sample data file must be added (seqno, name and gender).

Each field needs to be added using the toolbar Add Field icon, or by clicking **Add** > **Field**. Complete the following tasks for each field:

- Enter a Field Name.
- Select a Field Type.

Use the following procedures to add each field.

To add the ID field:

1. Select a Field Type of NUMCHAR (numeric character).
2. Enter the number of digits of Precision (2) and the Length (2) field. The value in the **Length** field must equal or be greater than the **Precision** value. These values indicate the maximum size of the input, allowing for leading blanks or other characters.
3. Click **OK**.

   The field is added to the record display.

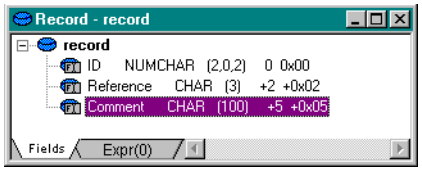
To add the Reference field:

1. Select a Field Type of CHAR (character).
2. Enter the number of digits of the Length (3) field.
3. Click **OK**.

To add the Comment field:

1. Select a Field Type of CHAR (character).
2. Enter the number of digits of the Length (100) field.
3. Click **OK**.

   After all the fields are complete, the record display looks like the following:



   Having completed the record we can now create a table.

Step 4. Add a Table

Use the following procedure to add a table.

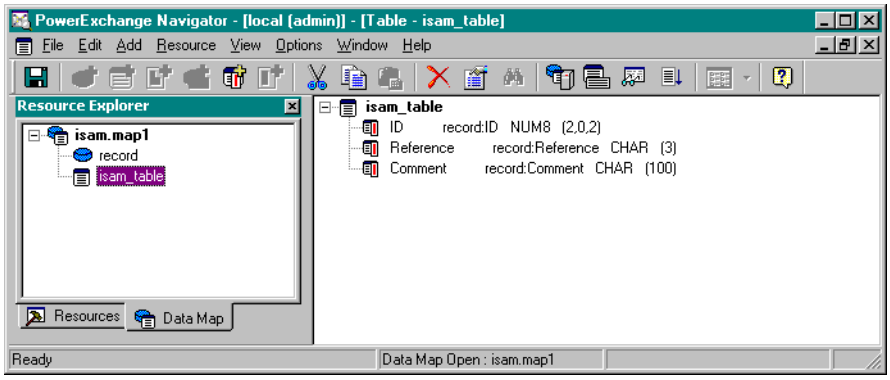
To add a table:

1. Right-click the data map name and click **Add Table**. This displays the Add Table dialog box. For this example, enter a **Table Name** of table1.

   In this example all the data from an input record is going to make up a row.

2. Click **OK**.

The table is created using all the fields in record.



This data map is now complete. It can be used to test our data source immediately. In this example we are not going to connect to a remote PowerExchange Listener, but go directly to the data file locally.

Step 5. Test the Data Map

Use the following procedure to test the data map.

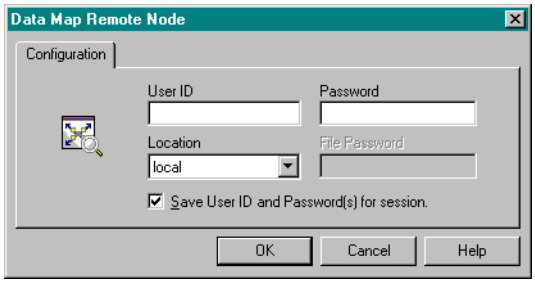
To test the data map:

1. In the Resource Explorer, double-click the data map.
2. On the Data Map tab, select a table. Then click **File > Database Row Test**.
3. In the message box that prompts you to send the data map to a remote location, click **Yes**.

   The Data Map Remote Node dialog box is displayed.
4. In the message box that prompts you to send the data map to a remote location, click **Yes**.

   The Data Map Remote Node dialog box is displayed.

   Make sure the Location is set to local as shown in the following figure:



5. Click **Yes**.
6. Click **OK**.
7. On the Database Row Test dialog box, click **Go**.

   Your source data appears.
8. In the Database Row Test dialog box, click **Go**.
9. Save the data map.
10. Close the Database Row Test dialog box.
11. Click **File > Close Resource**.
12. The isam.map1 data map appears in the **Data Maps** resource folder.

Creating MQSeries Data Maps

To move around data using MQSeries functionality, you must know the structure of the file, tables, and columns so that you can define a data map.

Typically the MQ data map must be column mapped to the data file that you wish to move with it. For example we may be moving a VSAM file that has been mapped using a VSAM data map. We must ensure that the VSAM data map and this MQ data map are matched.

Step 1. Add an MQSeries Data Map

Use the following procedure to add an MQSeries data map.

To add an MQSeries data map:

1. In the Name dialog box, perform the following actions:
   - Enter a **Schema Name**, such as mq1 and a Data Map Name, such as map1.
   - For MQSeries access, select the MQSERIES access method.
   - Clear the **Import Record Definitions** option. You specify record definitions later.
2. Click **Next**.
3. From the MQSeries Access Method dialog box, enter the name of the MQSeries Queue Manager and the queue name.
4. Click **Finish**.

Step 2. Add the Records

Use the following procedure to add the records.

To add the records:

1. Select **Properties**.

   Certain field formats are supplied including several date and timestamp formats. The timestamp selected previously redefines the 26 byte character field. The formats can be edited, however, and if the data is delivered without hyphens between the year, month and day delete these from the field format to match the field length and format found in the file.
2. Right-click the required data map and click **Add Record**. A blank Add Record dialog box appears. The name is for reference when building tables.
3. Enter the required record name, such as record.
4. Click **OK**.

   The Record panel appears.

Step 3. Add a Field

For example there may be three fields in the sample data file that must be added (seqno, name and gender).

Each field needs to be added using either the toolbar Add Field icon or through the menu option Add, Field. Complete the following tasks for each field:

- Select a Field Type.

  The details for each field that is to be added are as follows:

To add the ID field:

1. Select a Field Type of NUMCHAR (numeric character).
2. Enter the number of digits of Precision (2) and the Length (2) field. The Length field must be equal or greater than the Precision. This tells PowerExchange the maximum size of the input, allowing for leading blanks or other characters.
3. Click **OK**.

   The field is added to the record display.

To add the Reference field:

1. Select a Field Type of CHAR (character).
2. Enter the number of digits of the Length (3) field.
3. Click **OK**.

To add the Comment field:

1. Select a **Field Type** of CHAR (character).
2. Enter the number of digits of the Length (100) field.
3. Click **OK**.

Step 4. Add a Table

Use the following procedure to add a table.

To add a table:

1. Right-click the data map name and click **Add Table**. This displays the Add Table dialog box. For this example, enter a **Table Name** of table1.

   In this example all the data from an input record is going to make up a row.
2. Click **OK**.

   The table is created by using all the fields in record.

   You can use the data map to move data from its source to an MQSeries queue.
3. To save the data map, select **File > Close Resource**.

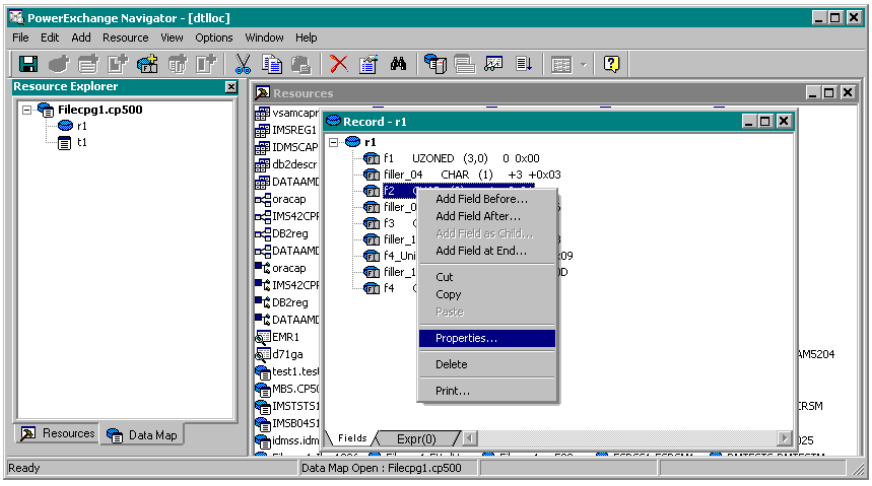
   The mq.map1 data map appears in the **Data Maps** resource folder.

Data Map Code Pages

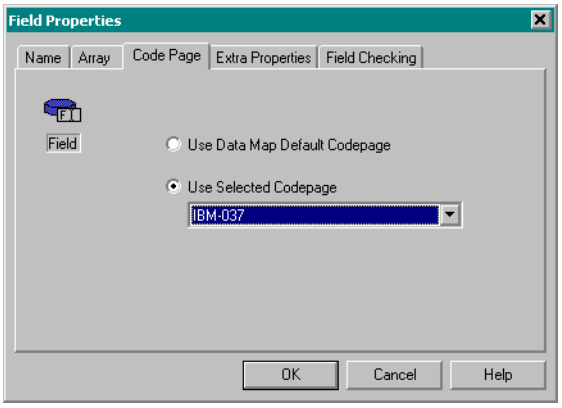
PowerExchange supports a wide range of code pages. Extended single and multi-byte support is available for DB2 on MVS and i5/OS, and NRDB access to VSAM and sequential files. To facilitate extended single byte and multi-byte support PowerExchange includes ICU software. Where the code pages pre-supplied with PowerExchange are not sufficient, the facility exists for the user to supply external code pages. Code page can be set at both the data map and field level.

The code page option is only active when the Encoding is explicitly defined as EBCDIC, ASCII HIEND and ASCII LOEND.

To specify a different code page at a field level, select a data map, open the record, then select properties of the required field.



The code page for the selected field can then be specified on the following screen:



Code page specified at the field level takes precedence over anything specified at the data map level.

PowerExchange Code Pages

The following table lists the standard code pages that are shipped with PowerExchange:

Code Page	Description
ISO-8859	Typically UNIX and Windows
IBM-037*	USA, Canada, Brazil
IBM-273*	Germany, Austria
IBM-277*	Denmark, Norway
IBM-278*	Finland, Sweden
IBM-280*	Italy
IBM-284*	Spain, Latin America
IBM-285*	United Kingdom
IBM-297*	France
IBM-424*	Modern Hebrew
IBM-500*	International
IBM-1047*	Latin 1/Open Systems
**Code pages prefixed with IBM- can handle data from MVS and i5/OS systems only.*	

The following table lists the default code page for each PowerExchange platform:

Operating System	Default Code Page
i5/OS	IBM-037
MVS	IBM-037
Windows	ISO-8859
UNIX	ISO-8859

You can set up to ten user code pages named USRCP*nn*, where *nn* is a value between 00 and 09. You set up user code pages to display or print specified hexadecimal values.

For more information about code pages and how to set up user-defined pages, see the *PowerExchange Reference Manual*.

CHAPTER 3

Importing Copybooks

This chapter includes the following topics:

Importing COBOL Copybooks

This chapter is intended to demonstrate how to import a COBOL Copybook through the PowerExchange Navigator and then source actual data. It contains the complete sequence of dialog boxes with the related text.

The key features demonstrated in this section are as follows:

- The data file is fixed-length records, 60 bytes long.
- Import of a COBOL copybook.
- Multiple record types, 01 for COBOL NAME_REC, 02 for COBOL ACCOUNT_REC.
- A hierarchical view of a flat file.
- An OCCURS depending on.
- The ability to generate multiple output rows from a single input row.
- Using date masking of COBOL POLICY_DATE.
- Use of a where clause to eliminate Donald Duck record.

Data File Input

File name: train3.dat, fixed 60 bytes long

```
00501Mickey Mouse        M3apple     orange    pear      .
005020000012345012-31-87                                 .
01001Shirley Temple      F1raspberry                     .
010020000000025607-04-57                                 .
02001John Wayne          M2pansy     daisy               .
03001Donald Duck         M0                              .
030020000000004501-12-66                                 .
04001Goofy               M1fox                           .
05001Greta Garbo         F3dog       cat       rabbit    .
050020000000091102-29-96                                 .
06001Ronald Reagan       M2horse     pony                .
060020000100000001-01-01                                 .
07001Bette Midler        F1wolf                          .
070020000064000112-07-41                                 .
```

Data File Input Viewed through the Copybook

```
005 01 Mickey Mouse         M 3 apple     orange    pear      .
005 02 00000123450 12-31-87                                   .
010 01 Shirley Temple       F 1 raspberry                     .
010 02 00000000256 07-04-57                                   .
020 01 John Wayne           M 2 pansy     daisy               .
030 01 Donald Duck          M 0                               .
030 02 00000000045 01-12-66                                   .
040 01 Goofy                M 1 fox                           .
050 01 Greta Garbo          F 3 dog       cat       rabbit    .
050 02 00000000911 02-29-96                                   .
060 01 Ronald Reagan        M 2 horse     pony                .
060 02 00001000000 01-01-01                                   .
070 01 Bette Midler         F 1 wolf                          .
070 02 00000640001 12-07-41                                   .
```

COBOL Copybook with OCCURS

train3.cob

```
01  NAME_REC.
04  ACCOUNT               PIC 9(3).
04  RECTYPE               PIC 9(2).
04  NAME                  PIC X(20).
04  SEX                   PIC X.
04  ITEMCT                PIC 9.
04  ITEMS OCCURS 3 DEPENDING ON ITEMCT PIC X(10).
04  FILLER                PIC XXX.
01  ACCOUNT_REC.
04  ACCOUNT               PIC 9(3).    04  RECTYPE               PIC 9(2).
04  AMOUNT                PIC 9(9)V99.    04  POLICY_DATE           PIC X(8).
04  FILLER                PIC X(36).
```

Output Row to be Constructed

- Account
- Name
- Sex
- Amount
- Policy date as a formatted date

Loading a COBOL Copybook

To load a COBOL copybook is a simple process of following the wizard provided. A copybook can be imported into an existing data map or a new one.

Step 1. Add a Data Map

Use the following procedure to add a data map.

To add a data map:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - In the Schema Name box, enter *xxxx*.
   - In the Data Map Name box, enter map2.
   - Select the Import Record Definitions (IRD) option to invoke the import process automatically.

Step 2. File Name and Record Format

Use the following procedure to enter a file name and record format.

To enter a file name and record format:

1. On the SEQ Access Method dialog box, browse through the supplied examples directory for the file name train3.dat and open the file. For example: C:\Program Files\Informatica\Informatica PowerExchange\examples\train3.dat.
2. Select **Fixed**.
3. Enter a Size of 60.
4. Leave the Encoding as Default.
5. Click **Finish**.

Step 3. Import Copybook Wizard

The Import Copybook wizard opens automatically.

To complete information in the Import Copybook wizard:

1. In Source, select **Local**.
2. In Source, make sure that the dialog box shows COBOL in the Type field.

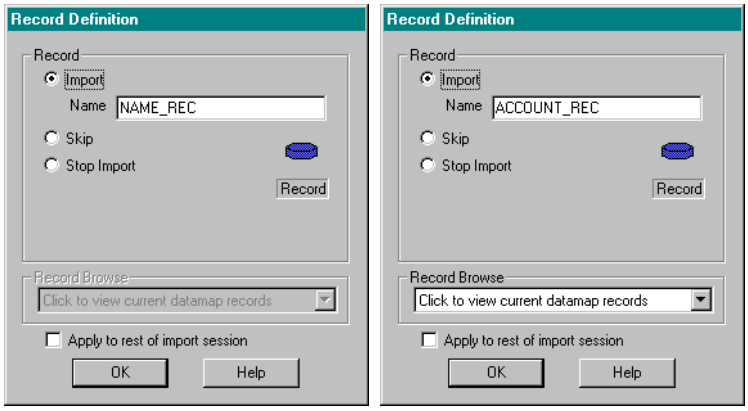
   The source is the node name where the copybook exists. PowerExchange can for instance go directly to MVS and retrieve the copybook from a PDS.
3. In Column Range, define the Start and End columns to be inspected. Typically, set these to 7 and 72 for IBM copybooks, but other ones might have data outside this range. The value entered for the Start column is recognized as the last comment column. The values should show 7 and 72.
4. Click **Next** to request the copybook file name. Browse for train3.cob in the examples directory and open it. For example: C:\Program Files\Informatica\Informatica PowerExchange\examples\train3.cob.
5. Click the **Preview** button to view the copybook file being imported. The Field Usage, Level, Original Name, and Picture from the Cobol or PL1 copybook are visible after the import has completed in the Extra Properties tab.
6. Close the Preview dialog box if viewed.
7. To display the Import Copybook - Configuration Details dialog box, click **Next**.
8. To display the Import Copybook Information dialog box, click **Finish**.

Note: The Import Copybook Configuration Details dialog box enables you to change the prompt parameters. To skip this dialog box, click **Finish** on the Import Copybook - Local COBOL Details dialog box. However, view this dialog box to confirm selected options.

The Import Copybook Information dialog box enables you to confirm the selected copybook options. If these are incorrect, click Cancel to reset them.

If the settings are correct, click **OK**.

The Record Definition dialog box appears for the two records to be imported. The record names as the same as the ones in the COBOL copybook.



- Select **Skip** to skip a record type.
- Select **Stop Import** to terminate the import.

9. In this example, because you want to import both record types, accept the defaults and click **OK** on both dialog boxes.

   The Copybook import dialog box appears.

   An Error/Log appears at the end of the import showing the outcome.

10. After successfully completing this task, close the Copybook Import dialog box.

    Note: If there is an error in the Copybook Message Log, you can double-click the message to display the relevant line number in the Cobol Import window.

Step 4. Change Record ID Value

Use the following procedure to change the record ID value.

To change the record ID value:

1. On the Data Map tab, click NAME_REC.
2. Right-click the RECTYPE field and click **Properties**.
3. Click Record ID Values and enter 01.
4. Click **OK**.
5. Repeat the process for ACCOUNT_REC with a Record ID Value of 02.

   Note: To review the original COBOL or PL1 field attributes from the Cobol copybook click the Extra Properties tab.

6. Click **OK**.

   Both RECTYPE field icons should be green.

Step 5. Add Table

Use the following procedure to add a table.

To add a table:

1. Right-click map2, and then click **Add Table**.
2. Enter a Table Name of both.
3. Right-click NAME_REC in the left dialog box and click **Add Record**.
4. Right-click ACCOUNT_REC in left dialog box and click **Add Record as Child**.
5. After creating the hierarchical definition, click **OK**.

Step 6. Row Test

Use the following procedure to perform a row test.

To perform a row test:

1. Click the Row Test icon.

   A message box indicates that the data map needs to be sent to a remote node.
2. Select **Don't warn** to eliminate the prompt.
3. Click **OK**.

   The Data Map Remote Node dialog box appears.
4. Make sure the Location is set to local.
5. Click **OK**.

   The Database Row Test dialog box appears.
6. Click **Go**.

   The data is retrieved from the data file and is displayed.

Step 7. Delete Unwanted Fields

Use the following procedure to delete unwanted fields.

To delete unwanted fields:

1. Close the Database Row Test dialog box.
2. Click the both table and delete the RECTYPE, ACCOUNT_REC_RECTYPE and ACCOUNT_REC_ACCOUNT fields.
3. To delete a field, right-click the appropriate field and click **Delete**.

Step 8. Add Field Format for Policy Date

Use the following procedure to add a field format.

Note: Be aware of the following:

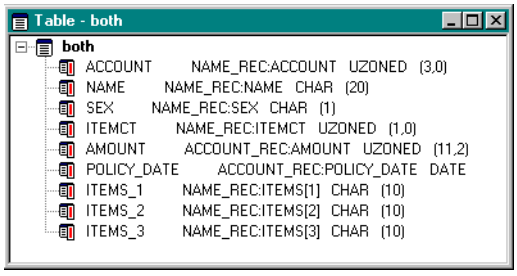
- The POLICY_DATE shows as an unformatted field, which you can format.
- The FILLER fields are normally excluded by default. However, in this example, you must delete them manually.

To add a field format:

1. Click the record ACCOUNT_REC.
2. Right-click POLICY_DATE field and click **Properties**.

   The Field Properties dialog box appears.
3. Click the arrow button to see the format options for Field Format.
4. Select Y2-MM-D2. Do not include spaces in the mask.

   The POLICY_DATE field type has changed from CHAR(8) to DATE.



5. Click the both table and then proceed with another Row Test.
6. On the Database Row Test dialog box, click **Go**.

   The following actions occur:

   - The century has now been added to the date.
   - The hyphens within the POLICY_DATE data are for display purposes only.
   - Donald Duck has zero items. Therefore, a where clause can be used to remove it.

Step 9. Use Where Clause to Eliminate Donald Duck

Use the following procedure to use a where clause.

To use a where clause to filter data:

1. Add where itemct > 0 to SQL statement.

   It should read:

   ```
   select * from xxxx.map2.both where itemct > 0
   ```
2. On the Database Row Test dialog box, click **Go**.

   The following actions occur:

   - Only seven rows are displayed.
   - Several NULL values are supplied in ITEMS field.
   - See the next step for an example of a single ITEM per output row (OCCURS DEPENDING ON).

Step 10. Single Item per Output Row

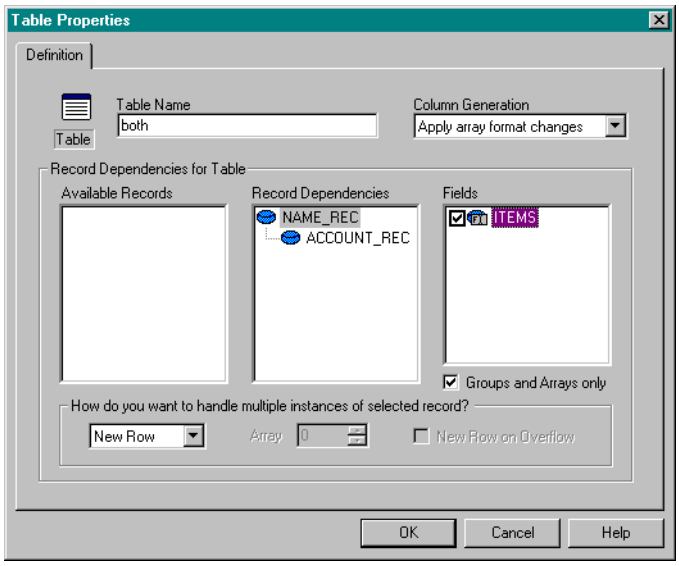
Use the following procedure to display a single item per output row.

To display a single item:

1. Close the Database Row Test dialog box.
2. Right-click the both table and click **Properties**.

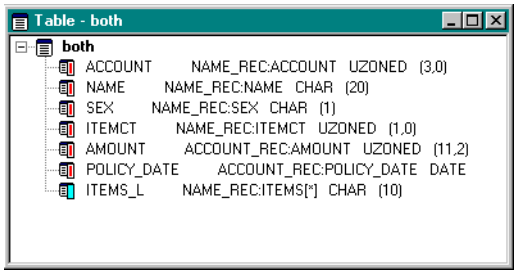
   The Table Properties dialog box appears.

Check the ITEMS box as shown in the following figure:



3. Click **OK**.

   The ITEM field icon has changed color and there is a single field whereas before there were three.



4. Perform a Row Test. On the Database Row Test dialog box, click **Go**.

   Note: Now three records exist for Mickey Mouse with a single ITEM per output.

5. Close the Database Row Test dialog box.

6. Click **File > Close** Resource.

Importing COBOL Copybooks Containing Redefines

This chapter is intended to follow “Importing Copybooks” on page 75, teaching you how to import a more complex COBOL Copybook through the PowerExchange Navigator and then to source the data. The input data source is an EBCDIC file, simulating data sourced from an IBM mainframe.

The key features demonstrated in this section are as follows:

- A fixed-length binary data file of EBCDIC data, 57 byte records, 58320 rows
- COBOL copybook with line numbers in columns 1-6 and comments in columns 73-80
- Redefines in copybook imported
- First record to be ignored
- Date masking on MASTER_DATE field

Data File Input

The file name is train5.dat. This file contains EBCDIC data after a single ASCII record.

COBOL Copybook

The file name is train5.cob.

```
00001 * TRAIN5 EXAMPLE COBOL COPYBOOK
00002   01 MASTER_REC.                                        COL 73-80
00003      05  ACCOUNT_NO             PIC X(9).               COL 73-80
00004      05  REC_TYPE               PIC X.                  COL 73-80
00004      05  AMOUNT                 PIC S9(4)V99 COMP-3.    COL 73-80
00005      05  BIN-NO                 PIC S9(8) COMP.         COL 73-80
00006 * REDEFINES OF BINARY FIELD TO CHARACTER
00007      05  BIN-NO-X REDEFINES BIN-NO   PIC XXXX.          COL 73-80
00008      05  DECIMAL-NO                  PIC S999.          COL 73-80
00009      05  MASTER-DATE.                                   COL 73-80
00010          10  DATE-YY            PIC 9(2).               COL 73-80
00011          10  DATE-MM            PIC 9(2).               COL 73-80
00012          10  DATE-DD            PIC 9(2).               COL 73-80
00013      05  ACCT-CODES             PIC X(10) OCCURS 3.     COL 73-80
```

Note: The comment lines in the copybook are treated as such and are ignored by PowerExchange.

Output Row to be Constructed

- account
- amount
- bin_no
- master date as single date column
- single row per account code

Loading the Copybook

Note: The skip records feature can be used, for example, to skip header records, which do not contain data. Range of this value is 0 to 2147483647.

Step 1. Add Data Map

Use the following procedure to add a data map.

To add a data map:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - Enter a Schema Name of *xxxx*.
   - Enter a Data Map Name of map3.
   - Select the Import Record Definitions box option.
3. Click **Next**.

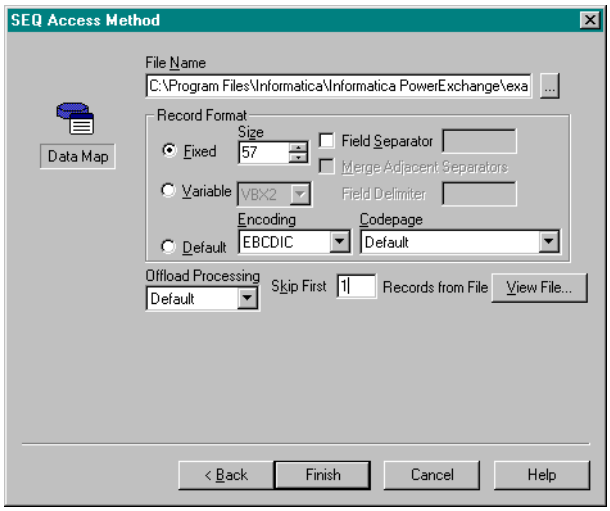
Step 2. File Name and Record Format

Use the following procedure to enter a file name and record format.

To enter a file name and record format:

1. In the File Name box, browse for train5.dat and open it.
2. Select **Fixed**.

3. Enter a record Size of 57.
4. For Encoding select EBCDIC.
5. Leave the Codepage as Default.
6. Skip First Records from File must be 1 for this example. The SEQ Access Method dialog box appears as follows:



7. Click **Finish**.

Step 3. Import Copybook

The Import Copybook dialog box appears.

To import a copybook:

1. In Source, select Local.
2. In Source, make sure that the dialog box shows COBOL in the Type field.

   The source is the node name where the copybook exists. PowerExchange can for instance go directly to MVS and retrieve the copybook from a PDS.
3. In Column Range define the Start and End columns that should be inspected. Typically these should be set to 7 and 72 for IBM copybooks, but other ones might have data outside this range. It should be stressed that the value entered for the Start column is deemed to be the last comment column. The values should show 7 and 72.
4. Click **Next**.
5. In the File Name box, browse for train5.cob and open it.
6. Click **Next**.
7. In the Configuration Details dialog box, click **Finish**.
8. In the Copybook Information dialog box, click **OK**.

Step 4. Define a Record

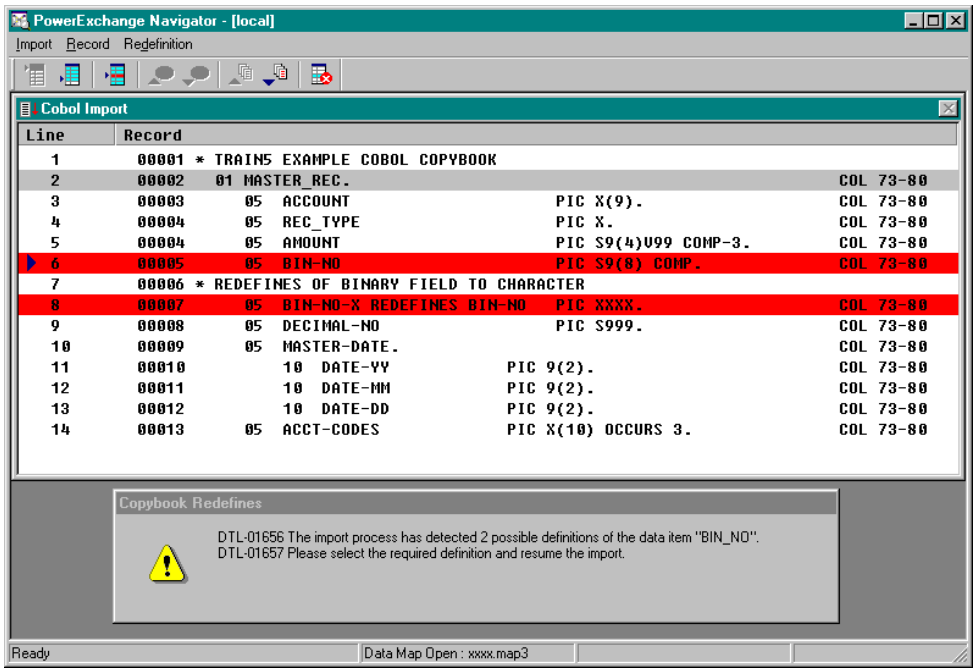
Use the following procedure to define a record.

To define a record:

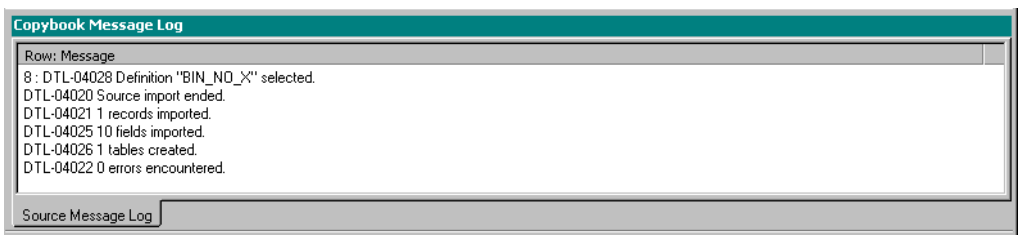
1. On the Record Definition dialog box, click **OK** on the MASTER_REC.

   The Copybook Import dialog box appears.

2. Use the Next Redefinition icon and Previous Redefinition icon on the toolbar to move the blue arrow on left to point at the line that you want to select.



3. Point at the PIC S9(8) field and click the Resume Import icon to resume the import. The following Error/Log Message should be displayed.



4. The first two lines in the log refer to record numbers in the Copybook Import dialog box.
5. Close the Copybook Import dialog box.

Step 5. Create Single Date Column

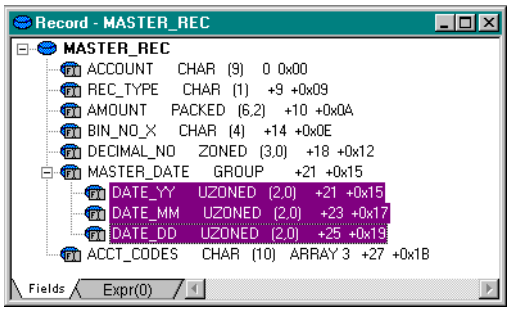
Use the following procedure to create a single date column.

To create a single date column:

1. Click the Data Map tab. Click MASTER_REC record, and click MASTER_DATE field.

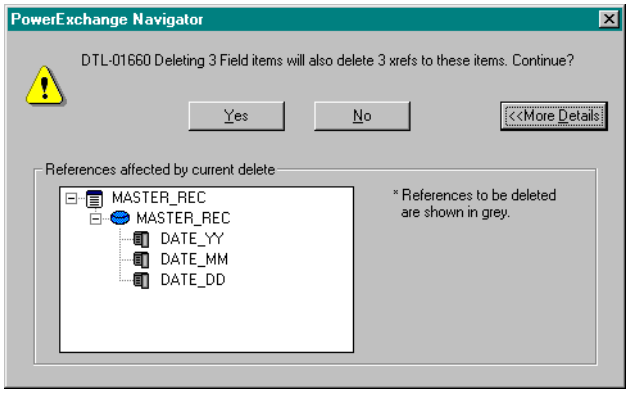
   Note: The COBOL copybook has a GROUP field and three individual ZONED fields, DATE_YY, DATE_MM and DATE_DD. However what is required on output is a single DATE column from this GROUP structure.

2. Highlight the three individual date fields. Click the DATE_YY field then hold down CTRL key and select the other two, as shown in the following window:



3. Click the Delete icon on the toolbar.

4. Click **More Details** to display field details.



5. Click **Yes**.

Step 6. Change Field Properties

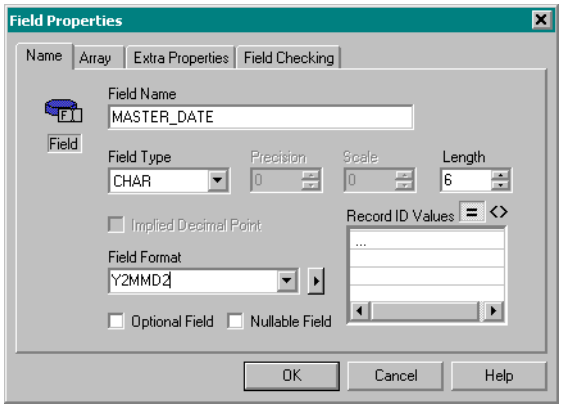
Use the following procedure to change field properties.

To change field properties:

1. Right-click the MASTER_DATE field and click Properties.
2. Change the Field Type to CHAR, and Length to 6.
3. For the Field Format, enter Y2MMD2 or select Y2-MM-D2 from the Field Mask and delete the hyphens.

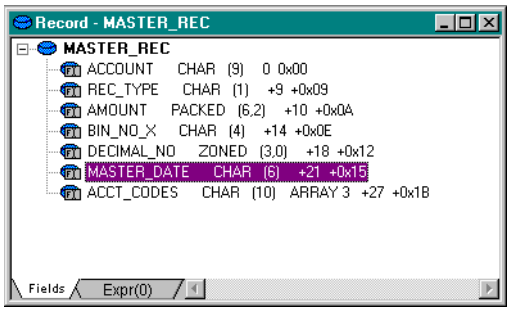
   Note: Do not select the Field Mask without deleting the hyphens otherwise you increase the number of characters to 8.

   The completed Field Properties dialog box appears.



4. Click **OK**.

   The MASTER_REC record is displayed.



Step 7. Add Column

Use the following procedure to add a column.

To add a column:

1. Click MASTER_REC table.
2. Click the Add Column icon.
3. In the Add Column dialog box, enter a Name of new_date.
4. Select **MASTER_REC:MASTER_DATE** from the Base Field pull-down list.
5. Click **OK**.

Step 8. Row Test

Use the following procedure to perform a row test.

To perform a row test:

1. Click MASTER_REC table and click the Row Test icon.
2. In the Database Row Test dialog box click **Go**.

   The row test results appear.
3. Close the Database Row Test dialog box.
4. Click **File > Close Resource**.

Importing a PL/1 Copybook

Introduction

In this chapter a PL/1 Copybook is imported instead of a COBOL Copybook. The Access Method used is (SEQ). The data file being sourced is train5.dat and it is used in “Importing COBOL Copybooks Containing Redefines” on page 81; this section shows the PL/1 import.

Data File Input

The file name is train5.dat. This file contains EBCDIC data after a single ASCII record. The only record that displays properly with a text editor is the ASCII record which reads: This is a bogus first record which should be skipped.

PL/1 Loading

The file name is train5.pl1.

```
DECLARE 1 MASTER_REC,
2 ACCOUNT                      CHAR(9),
2 REC_TYPE                     CHAR(1),
2 AMOUNT                       FIXED DECIMAL(6,2),
2 BIN_NO UNION,
3 BIN_NO_N                  FIXED BINARY(16),
/* REDEFINES OF BINARY FIELD TO CHARACTER */
3 BIN_NO_X                  CHAR(4),
2 DECIMAL-NO                   PIC "S999",
2 MASTER-DATE,
3 DATE-YY             PIC "99",
3 DATE-MM             PIC "99",
3 DATE-DD             PIC "99",
2 ACCT-CODES              CHAR(10) DIM(3);
```

Procedure

Note: A PL/1 Copybook can be a substitute for a COBOL copybook.

Step 1. Add Data Map

Use the following procedure to add a data map.

To add a data map:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - Enter a Schema Name of *xxxx*.
   - Enter a Data Map Name of map5.
   - Select the SEQ access method.
   - Select the Import Record Definitions box option.
3. Click **Next**.

Step 2. Enter File Name

Use the following procedure to enter a file name.

To enter the file name:

1. In the SEQ Access Method dialog box, browse to the train5.dat file.
2. Select **Fixed**.
3. Enter a Size of 57.
4. In the **Encoding** list, click **EBCDIC**.
5. Click **Finish**.

Step 3. Import Copybook

The Import Copybook wizard opens automatically.

Use the following procedure to import the copybook.

To import the copybook:

1. In Source, select **Local**, which is the node name where the member exists.
2. In Source, select PL/1 from the Type pull-down box. For PL/1, the column range automatically defaults to 1 and 72.

3. Click **Next**.

   You are prompted for the file name of the PL/1 copybook.

4. Browse to the train5.pl1 file.

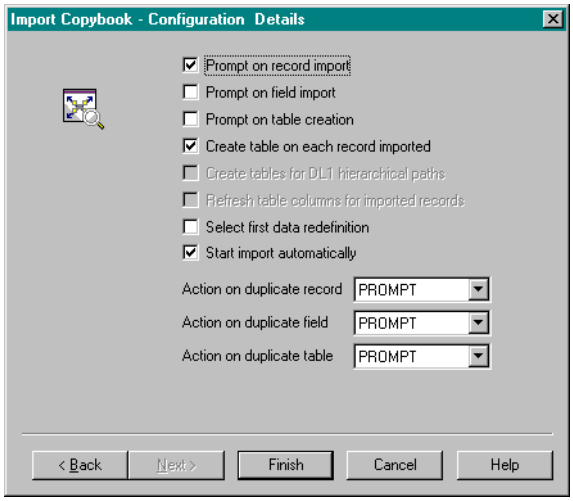
5. Click **Next**.

Step 4. Configuration Details

Use the following procedure to review configuration details.

To review configuration details:

1. A dialog box requesting what actions to be taken when loading the copybook appears.



2. In the Configuration Details dialog box, click **Finish**.

3. In the Copybook Information dialog box, click **OK**.

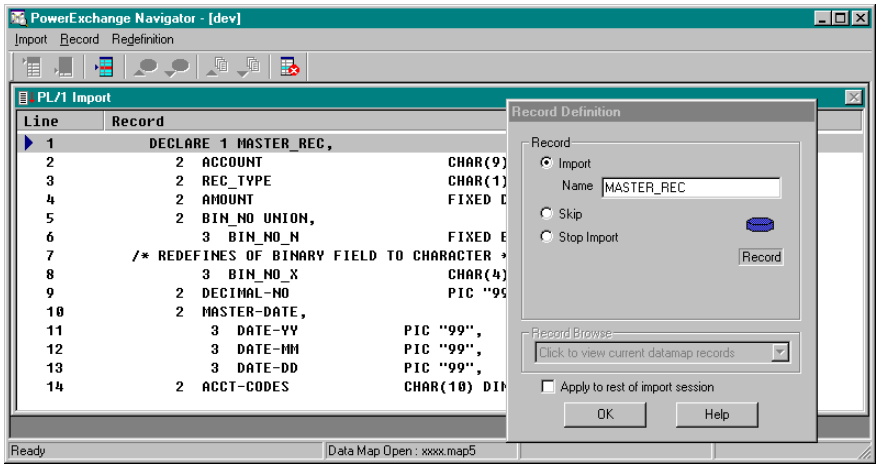
Step 5. Record Definition Dialog Box

Use the following procedure to define a record.

To define a record:

1. In the Record Definition dialog box, click **OK** for the MASTER_REC.

   The following dialog box appears:



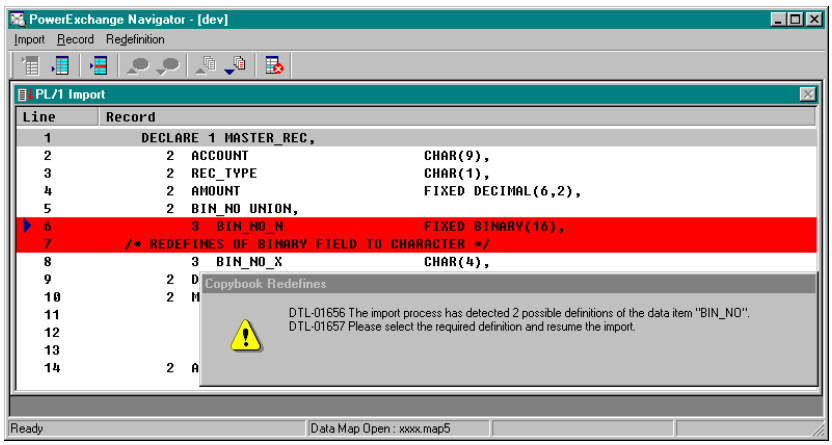
Accept and skip records. Skipping a record causes all entries to be skipped until the next DECLARE 1 statement is found, or the import is stopped.

Step 6. Select Redefinition Line

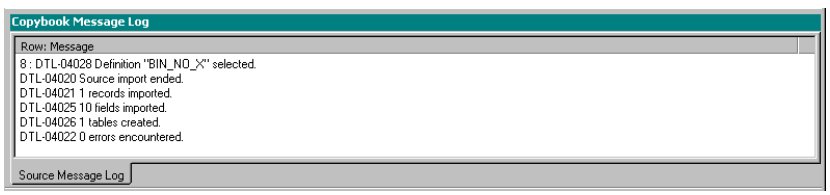
Use the following procedure to select a redefinition line.

To select a redefinition line:

1. There is a redefinition and the screen following appears, with the redefinitions in red.



2. Use Next Redefinition icon and Previous Redefinition icon on the toolbar to move the blue arrow on left to point at the line that you want to select.
3. Point the blue arrow at the line containing the Fixed Binary (16) field, which is line 6 and click the Resume Import icon to resume the import. The following Error/Log Message should be displayed.



Note: The first two lines in the log refer to record numbers in the PL/1 Import dialog box.

4. Close the PL/1 Import dialog box.

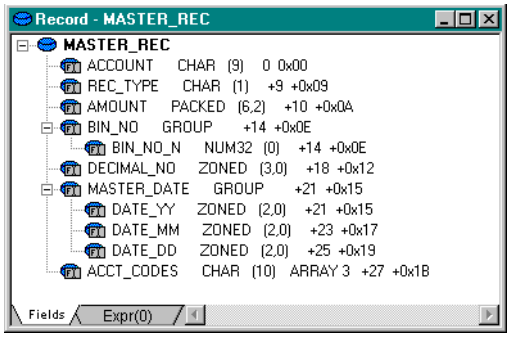
Step 7. View Record and Table

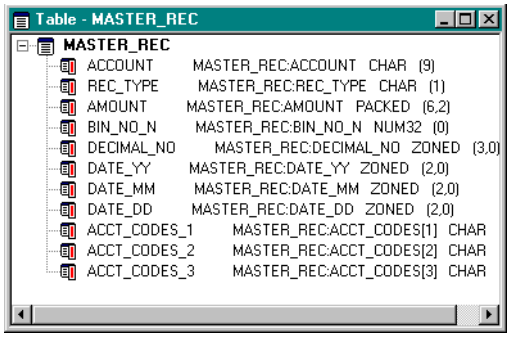
Use the following procedure to view a record and table.

To view a record and table:

1. Click MASTER_REC record and table to view the fields and columns that have been imported. Compare the field and table structures imported, relative to the COBOL Copybook import. The contents should be effectively identical.

The COBOL version shows BIN_NO as a group item, which has no impact.





2. Click **File > Close Resource**.

Importing a DDS Copybook

Use the following procedure to import a DDS copybook.

Introduction

In this example, a DDS copybook is imported instead of a COBOL or PL/1 copybook. The access method used is SEQ.

Procedure

Note: A DDS copybook can substitute for a COBOL or PL/1 copybook.

Step 1. Add Data Map

Use the following procedure to add a data map.

To add a data map:

1. Right-click **Data Maps** and click **Add Data Map**.
2. In the Name dialog box, perform the following actions:
   - Enter a Schema Name of DDS.
   - Enter a Data Map Name of map1.
   - Select the SEQ access method.
   - Select the Import Record Definitions box option.
3. Click **Next**.

Step 2. File Name

Use the following procedure to enter a file name.

To enter a file name:

1. In the SEQ Access Method dialog box, enter the file name that you want to map. For example, on an i5/OS system, you might enter STQA/IFITEST1.
2. Select **Default**.
3. In the **Encoding** list, click Default.
4. Click **Finish**.

Step 3. Import Copybook

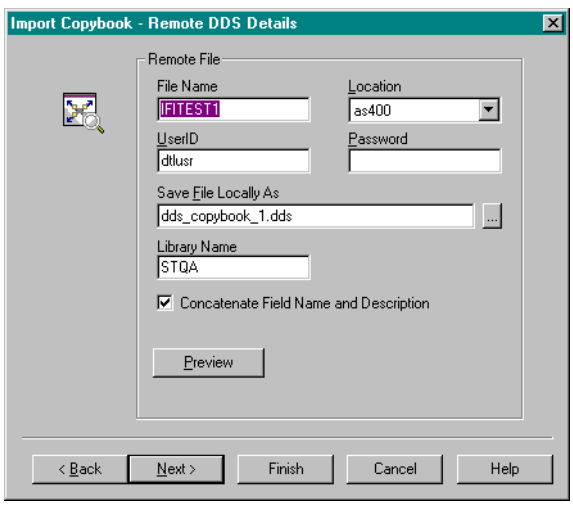
The Import Copybook wizard opens automatically.

Use the following procedure to import the copybook.

To import the copybook:

1. In the Import Copybook - Source Details dialog box, select Remote.
2. In the **Type** list, select DDS.
3. Click **Next**.

   The Import Copybook - Remote DDS Details dialog box appears.



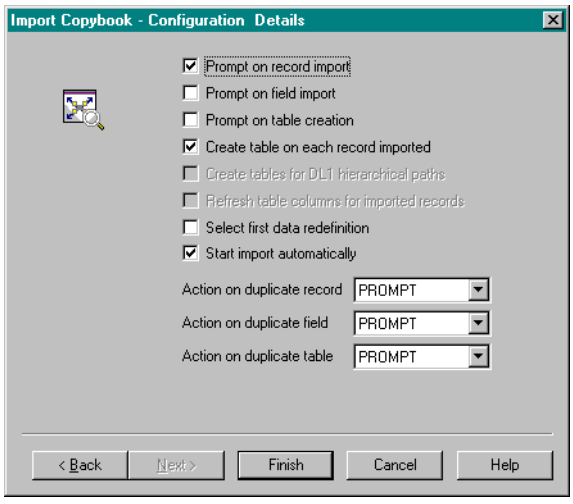
4. Enter the Remote File details.
5. Click **Next**.

Step 4. Configuration Details

Use the following procedure to review configuration details.

To review configuration details:

1. The Import Copybook - Configuration Details dialog box appears requesting what actions to be taken when loading the copybook.



   The options are very straightforward, requesting prompts in various situations and actions to be taken if duplicate record, fields and tables are detected.

2. In the Configuration Details dialog box, click **Finish**.
3. In the Import Copybook Information dialog box, click **OK**.

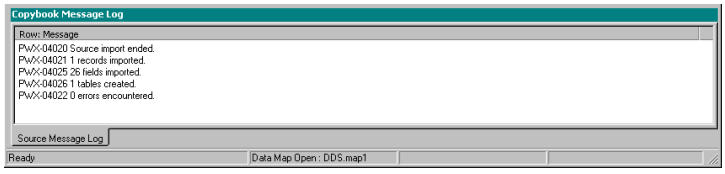
Step 6. Define a Record

Use the following procedure to define a record.

To define a record:

1. In the Record Definition dialog box, accept or skip records until the import stops or finishes normally.

   The following Message Log displays.



   Note: You need to review the log to ensure that the import has worked as expected.

2. Close the DDS Import dialog box.

Step 7. View Record and Table

Use the following procedure to view a record and table.

To view a record and table:

1. To view the fields and columns that have been imported, click the IFITEST1 record and table.
2. To close the data map, click **File > Close Resource**.

Importing Copybooks for IMS Segments in a Data Map

Import a COBOL copybook for each IMS segment added to a data map. Importing an IMS DBD source adds segments, key fields and search fields from the IMS database and creates tables for a data map. After you import the IMS DBD source, importing a copybook for each segment overlays each segment with its COPYLIB. This redefines the data map while maintaining the hierarchical metadata information for the database.

Before you import a copybook to an IMS data map, import the IMS DBD source. For more information, see "Importing the IMS DBD Source into an IMS Data Map" on page 51.

To import a copybook for IMS segments in a data map:

1. On the Data Map tab, select an IMS segment and click **File** > **Import Copybook**.
2. Complete the following fields on the Import Copybook - Source Details dialog box:

Field	Description
Source	Indicates the copybook is local which means the data has been downloaded and stored locally, or remote which means the data is downloaded from a remote location such as MVS. For instance, PowerExchange can go directly to MVS and retrieve a copybook from a PDS.
Type	Select the type of import to perform: COBOL.
Column Range	Defines the start and end columns to inspect.
Last Import	Click to show the specification to which the copybook is imported. The current specifications are those of the previous import.

3. If you selected **Local** as the **Source** option, continue to the next step.

   If you selected **Remote** as the **Source** option, go to step 5.
4. In the Import Copybook - Local Cobol Details dialog box, complete the following fields:.

Field	Description
File Name	File name of the copybook to import.
Preview	Click to preview the copybook.

5. Click **Next**.
6. In the Import Copybook - Remote Cobol Details dialog box, complete the following fields:

Field	Description
File Name	Copybook file name and member.
Location	Location of the copybook file.
UserID	MVS user ID if security has been implemented.
Password	MVS password if security has been implemented.
Name	Name of the file that the copybook is to be saved as when it is retrieved from the remote platform.
Name Browse	Click to browse for the file.
Preview	Click to preview the copybook.

7. Click **Next**.

8. In the Import Copybook - Configuration Details dialog box, complete the following fields:

Field	Description
Prompt on record import	When a record is created, prompt for a record name showing the first name found.
Prompt on field import	When a field is created, prompt for a name showing the name found.
Prompt on table creation	When a new table is created, prompt for a table showing the default name.
Create table on each record imported	Creates a table if a record is imported.
Create tables for DL1 hierarchical paths	Not available for copybook import.
Select first data redefinition	Select so that if a REDEFINES clause is found, the first data definition is automatically used.
Start import automatically	Select to automatically start the import.
Action on duplicate record	Select the action to take if a duplicate segment is found: PROMPT, provide a UNIQUE NAME, OVERWRITE it, or SKIP it.
Action on duplicate field	Select the action to take if a duplicate field is found: PROMPT, provide a UNIQUE NAME, OVERWRITE it, or SKIP it.
Action on duplicate table	Select if a table with the same name is found: PROMPT, provide a UNIQUE NAME, OVERWRITE it, or SKIP it.

9. Click **Finish**.
10. Review the selected options.
11. If the information is correct, click **OK**.

    If you chose to be prompted during the import, continue to "Using the Import Prompts" on page 94.

    If you did not choose to be prompted, review the data that is imported.

Using the Import Prompts

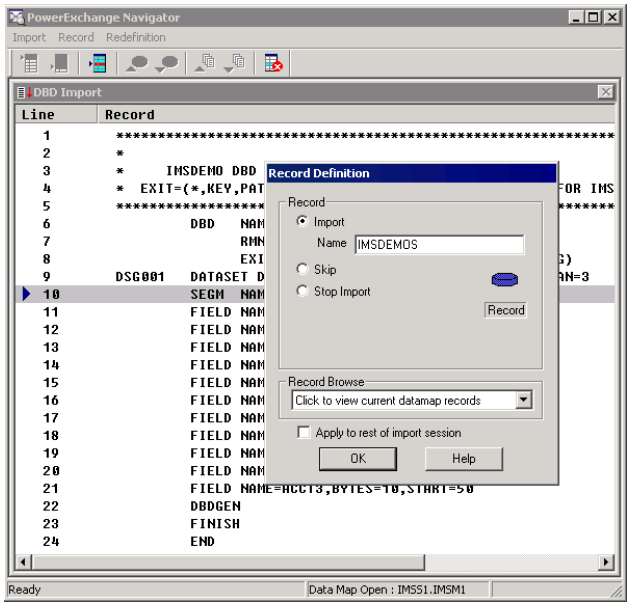
When you complete the Import Copybook Options, you indicate whether or not prompts should be provided as each item is imported from a DBD source or copybook.

If you choose to be prompted, use the Record Definition options to complete import prompts and step through each item that is being imported. During this process, you determine which segments to import or skip. You can also specify how to manage duplicate data during a copybook import.

To specify items to import:

- Complete the Record Definition options to step through each item as a DBD source or COBOL copybook is imported.

  Or, if you are starting the import from the DBD or Copybook Import view, select **Import > Start** to access the Record Definition options.



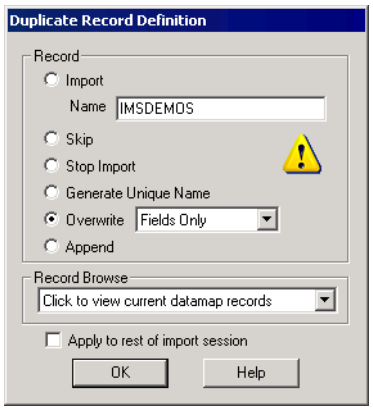
The following table describes the Record Definition options:

Option	Description
Import	Select the item to import it.
Name	Name of the item to import.
Skip	Select to go to the next item.
Stop Import	Select to stop the import process.
Record Browse	Specifies the record to view and apply the options to.
Apply to rest of import session	Applies the options you select to the rest of the items you import.

To replace fields in a segment when you import a copybook:

1. In the Record Definition options, change the name to match the existing segment name.
2. Click **OK**.

3. In the Duplicate Record Definition option, select **Overwrite** and then select **Fields Only**:



4. Complete the import.
5. On the Data Maps tab of the Resource Explorer, right-click the relevant table and select **Properties**.
6. Right-click the table in the **Record Dependencies** list and select **Properties**.
7. In the **Column Generation** list, select **Refresh with missing columns**
8. Click **OK**.

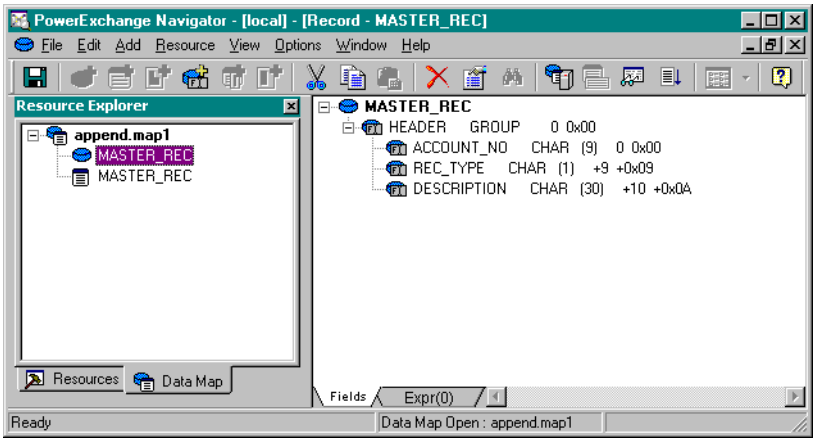
Creating Records from Multiple Copybooks

This section describes how to import multiple copybooks through the PowerExchange Navigator. For more information, see “Importing Copybooks” on page 75.

It is assumed that you have already created a data map using a single copybook. The following instructions show you how to import a second copybook and append its details to an existing record.

Adding a Second Copybook to the Record Definition

The starting point is a data map that has been created from a copybook, as shown.



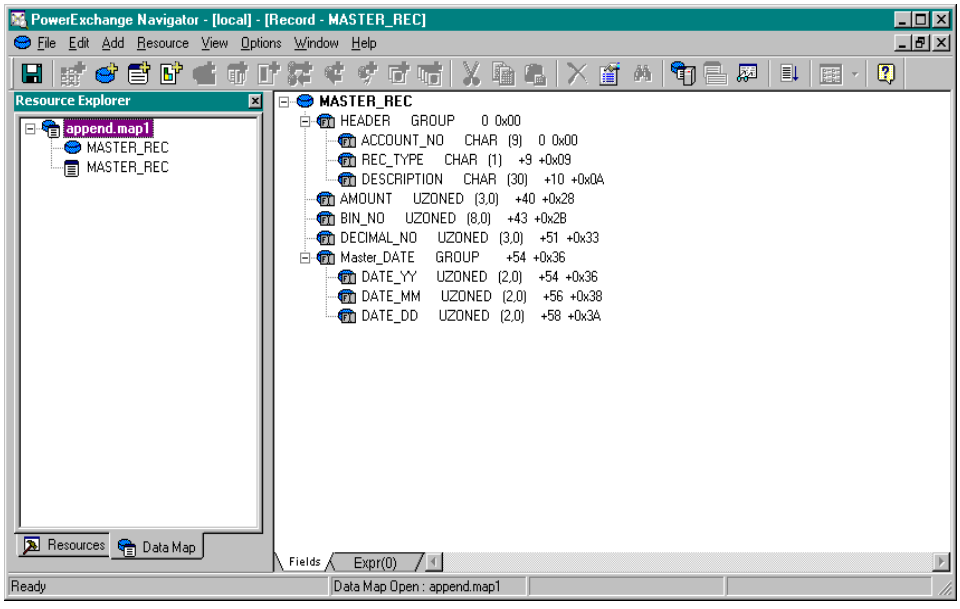
To add a second copybook to a record definition:

1. Select the record in the Resource Explorer.
2. Select **File > Import Copybook**.

   The Import Copybook - Source Details dialog box appears.
3. Click **Next**.
4. In the File Name box, enter the path and filename of the copybook you want to import.
5. Click **Next**.
6. Click **Finish**.
7. Click **OK**.
8. Click **OK**.
9. Select the MASTER_REC record name from the Record Browse pull-down list.
10. Select the **Append** option.

    This adds the details to the current record.
11. Click **OK**.
12. Select the Overwrite option to replace the table with the new definition.
13. Click **OK**.
14. Click **OK**.

    The MASTER_REC record appears as follows:



CHAPTER 4

Managing Data Maps

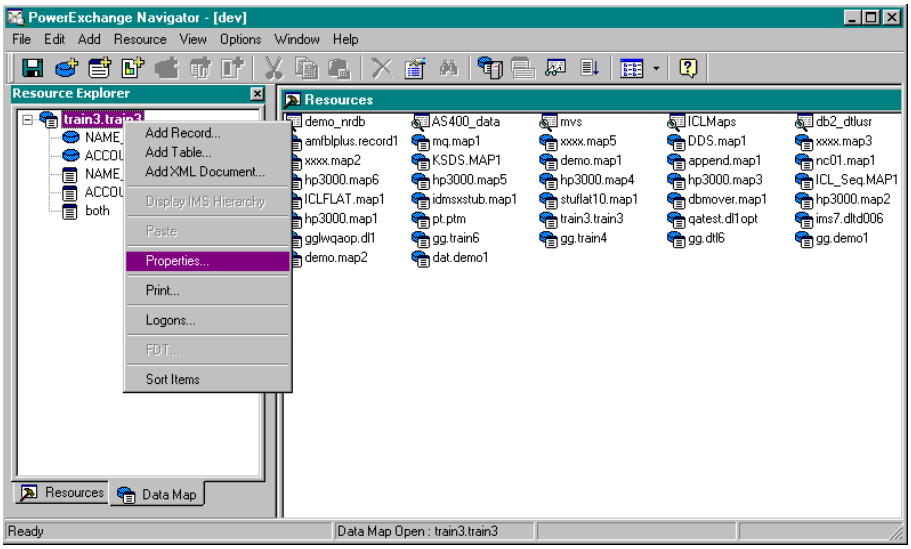
This chapter includes the following topics:

Data Checking

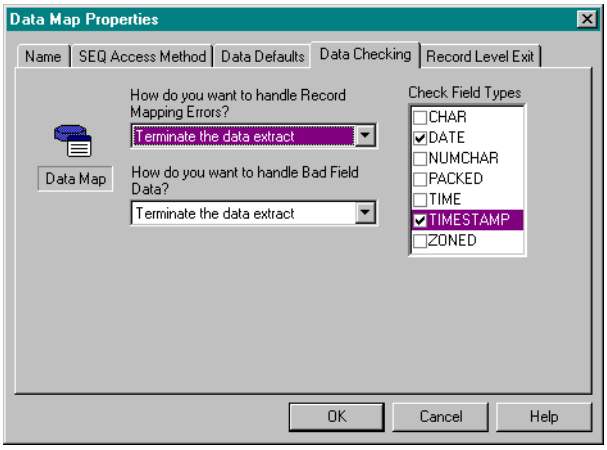
The quality of data to be extracted may not be consistently good, leading to failures during data selection and loading. PowerExchange supplies functionality to resolve bad data issues, providing two levels at which bad data can be handled. Record and field handling options can be defined at the Data Map level.

Note: Data Checking fields on every record use additional processing effort and should be used carefully.

To initiate Data Checking and cleansing right-click the Data Map name and select Properties.



From the resulting Properties screen select the Data Checking tag to progress to the following screen.



It is on this screen that the datatypes to be checked are selected and the actions to be taken on return of data mapping and field data issues are determined.

If errors are found at the record level, such as OCCURS DEPENDING ON value greater than the maximum permitted, you can either terminate the complete extract or skip the row and a message is inserted into the PowerExchange log file. It is good practice to inspect the source PowerExchange log file when initially extracting data with the second option, to determine if any data quality issues exist.

If you want specific field types checked select the correct box. The choices are as follows: CHAR, DATE, NUMCHAR, PACKED, TIME, TIMESTAMP and ZONED. The following table lists the validation for these fields:

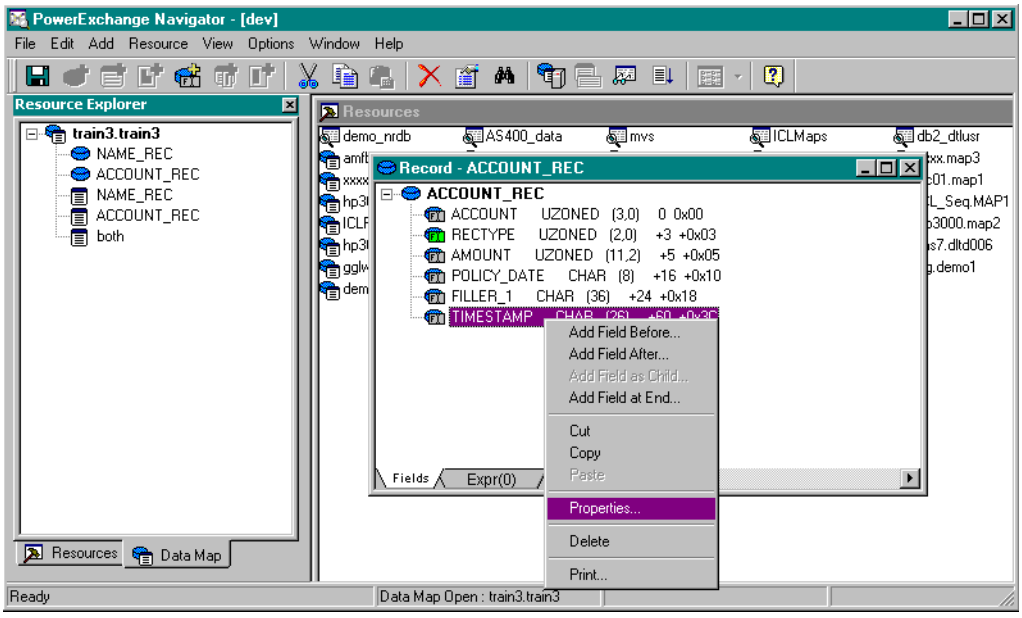
Field Type	Validation
CHAR	Standard alphanumeric rules, 0-9, A-Z, a-z and greater than or equal to space (EBCDIC and ASCII)
DATE	Date format, defined within PowerExchange.
NUMCHAR	0-9, +,-, decimal point, comma, $, /, *, space.
PACKED	Standard packed decimal format (S9 COMP-3).
TIME	Time format defined within PowerExchange.
TIMESTAMP	Timestamp format defined within PowerExchange.
ZONED	Zoned numeric allowing sign and 0-9.

Several options exist for handling bad field data:

- Terminate the data extract.
- Replace according to data defaults. For CHAR or DATE type fields this replacement value is taken from the value in the Data Defaults sheet (selected from the same screen the Data Checking screen is selected from). The value can be inserted in every character position in the field or just the first character. Numeric fields is set to zeros.
- Skip row and write log file msg. Skip the row, insert a message into the log file, and continue.
- Set to NULL. The invalid field is set to NULL.

The Date, Time and Timestamp validation is carried out against the PowerExchange defined Date, Time and Timestamp datatypes. To define the format of these datatypes the record must be edited.

Right-click the required column within the record.



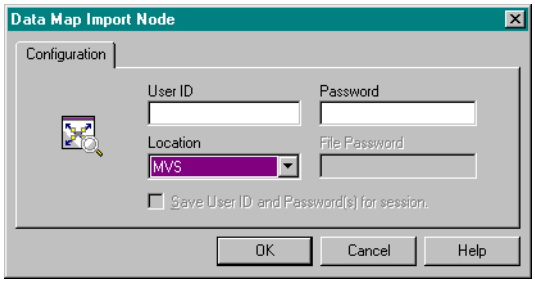
Enter a Field Name.

Importing Exported Data Maps

It is possible to use data maps that exist on other computers on your network. You can use them on a client by selecting the data maps from a remote computer where they have previously been saved.

To import exported data maps:

1. To import one or more data maps, click File > Exported Data Maps right-click Exported Data Maps.
2. Select the remote node from which to download the data maps.



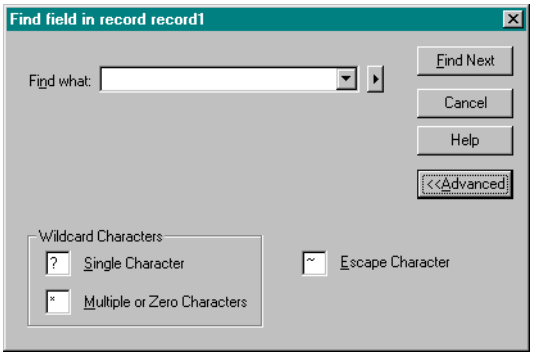
3. Click **OK**.
4. On the Exported Data Maps dialog box, select the data maps that you wish to import and click Import. The selected data maps are added to your data maps list and the Exported Data Maps dialog box is removed.

   Delete any data map by selecting the data map and clicking Delete.

Searching in Data Maps

There is a Search Facility for fields within records, or columns within tables. The facility can also be used with the Metadata Display. The facility can be operated in a simple or advanced mode. This facility is useful when a large number of fields, columns or tables are in the display. To perform a search, place the cursor on the element to be searched and use the toolbar Binoculars icon or click Edit > Find. For more information about Metadata Display, see “Search Facility for Metadata Display” on page 111.

Two search dialog boxes are available: Find Field in record and Find column in table. They both have similar functionality.



Clicking on the right arrow, the current wildcard characters that could be used in a search can be revealed.

A wildcard search change capability is also offered by clicking Advanced. This permits the Wildcard Characters and the Escape Character to be changed.

The Wildcard Characters (? and *) are available in all advanced searches. These characters were chosen, as they are unlikely to be used in table, field and column names. If they are being used, precede them with the Escape Character (~). For example a request for tab* would list all tables beginning with tab, whereas a request for tab~* would only list the table that was named tab*.

A search on item*, such as item + any other character on the records, items, and item_ct finds both of those fields.

However a search using item?, such as item + one other character, only finds items.

Note: View a list of previous searches by clicking on the down arrow beside the Find What box.

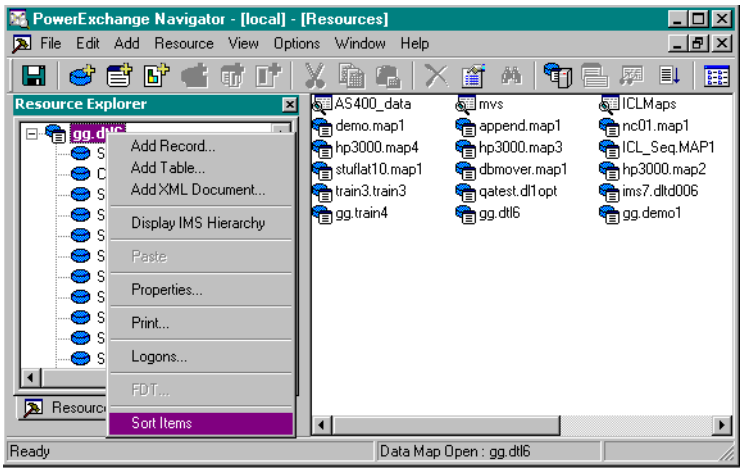
Sorting Data Map Records and Tables

When a data map has been created, the resulting data map may contain numerous records and tables. The records and tables are separated out and displayed as records and then tables, in the order in which they were added. To make it easier to select a particular record or table a sort by name feature is available.

To sort data map records and tables:

1. Select the data map from the list of data maps.
2. In the Resource Explorer, right-click the data map name.

3. Select Sort Items.



This causes the tables and records to be displayed in name order. This sort is temporary and upon selecting the data map the next time the records and tables revert to the original order.

File List Processing

File List Processing provides a facility whereby sequential files can be concatenated and processed as if all the data were in one file.

All the file names to be concatenated are specified in a special Filelist file. This file resides on the same platform as the data files that are being sourced.

The data map can then be processed in the normal way, such as from an ODBC call.

Restrictions

- File List Processing is available for SEQ(uential) or ESDS files and for TAPE file access.
- Only Read Mode is supported.

ODBC

File List Processing can be activated by means of an ODBC escape sequence DTLFILELIST and the Filelist definition file specified with DTLDSN.

For more information, see the *PowerExchange Reference Manual*.

Filelist File

All the file names to be concatenated are specified in a special Filelist file.

- This file must reside on the same platform as the data files that are being sourced.
- One file should be specified on each line.
- Blank lines ignored.
- Lines beginning /* are treated as comments and are ignored during processing.
- Blanks at the beginning of lines are ignored.

- The specified file name in a Filelist file is all the characters between the leading and trailing blanks on a line. Please note that this means that MVS files with sequence numbers set to ON include the sequence number as part of the file name.
- Whole filelist file is checked for syntactical errors ahead of any data source file reading but references to missing files or read errors are handled when that particular file is read. Process terminates at the point the error is found.
- Any invalid file names given in the Filelist file appear in their entirety in hex in the PowerExchange Listener log output to aid in debugging.

Example Filelist Files

Windows/UNIX

```
c:\pwx\data\filelist1.dat
c:\pwx\data\filelist2.dat
c:\pwx\data\filelist3.dat
```

MVS

```
****************************** Top of Data ******************************
000001 /* FILE LIST PROCESSING DEFINITION FILE
000002 /*
000003 DTLUSR.V522.V1.DEMO.ESDS1
000004 /*
000005 DTLUSR.V522.V1.DEMO.ESDS2
000006 /*
000007 DTLUSR.V522.V1.DEMO.ESDS3
000008 /*
****************************** Bottom of Data ***************************
```

Example Scenario

To demonstrate the use of File List processing the following simple example scenario uses a file list definition file on the computer that references three locally held sequential data files through the PowerExchange local mode.

Example Filelist Definition File

```
c:\pwx\data\filelist1.dat
c:\pwx\data\filelist2.dat
c:\pwx\data\filelist3.dat
```

Data Map

A simple data map was created to map the data. For more information about creating this data map, see “Example 1. Single-Record Data Map” on page 19.

However, instead of mapping to a single data file, such as demo1.dat, you check the File List Processing check box on the SEQ Access Method dialog box and enter the full file name and path of the Filelist file into the File Name box.

Row Test Results

The data held in these three files is concatenated in the output results.

```
1,Mickey Mouse,M
2,Shirley Temple,F
3,John Wayne,M
4,Donald Duck,M
5,Greta Garbo,F
6,Mickey Mouse,M
7,Shirley Temple,F
8,John Wayne,M
9,Donald Duck,M
10,Greta Garbo,F
11,Mickey Mouse,M
12,Shirley Temple,F
13,John Wayne,M
14,Donald Duck,M
15,Greta Garbo,F
```

CHAPTER 5

Personal Metadata

This chapter includes the following topic:

Personal Metadata Overview

In addition to using PowerExchange Navigator to build data maps and command sets, you can use it to access metadata for sources and targets. After retrieving the metadata, you can use the row test feature to view data. The row test feature enables you to discover the metadata that is present. Row test works for relational and nonrelational sources and is a powerful tool for viewing remote sources and targets.

Creating a Personal Metadata Profile

To use this facility, you define a Personal Metadata Profile. This stores the format of the metadata request, so that you can return to it later, reuse or modify it, and run it again. An example profile has been shipped with PowerExchange called demo_oracle.

Start by setting up a Metadata Profile to access the demo nonrelational data, which is also shipped with PowerExchange.

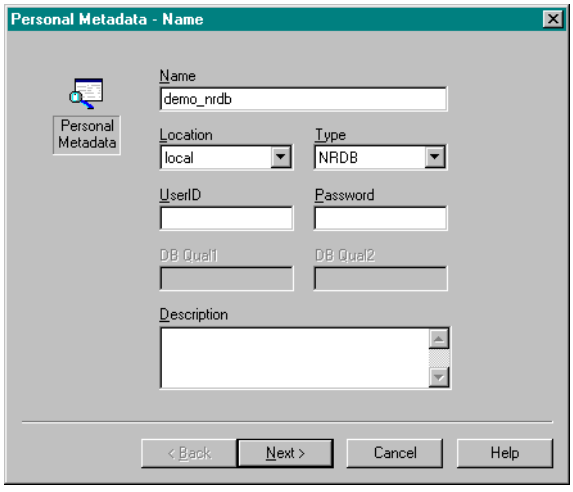
Step 1. Add Personal Metadata Profile

Use the following procedure to add a personal metadata profile.

To add a personal metadata profile:

1. Go to the Resources tab.
2. From the menu, click **Add** > **Personal Metadata**.

The Personal Metadata dialog box is displayed:



The dialog box contains the following fields:

- The Name is the resource name.
- The Location is wherever the resource is to be found.
- There is a drop down list for Type and this impacts on the dimmed areas on the dialog box. A number of optional fields are then displayed according to Type.
- The user ID and password may be required depending upon the SECURITY setting in the PowerExchange configuration file. If the password is blank, the Navigator prompts for the password to be entered as required. This prevents the password being stored and potentially causing password violations at a later date.

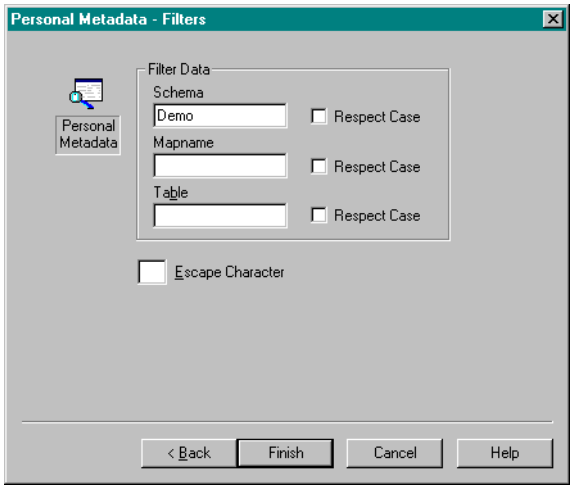
3. In this example, enter or select the following information:
   - **Name**: demo_nrdb
   - **Type**: NRBD
   - **Location:** local
4. Click **Next**.

Step 2. Enter Personal Metadata Filters

The Personal Metadata - Filters dialog box enables you to enter information that further qualifies the dtldescribe syntax.

To enter personal metadata filters:

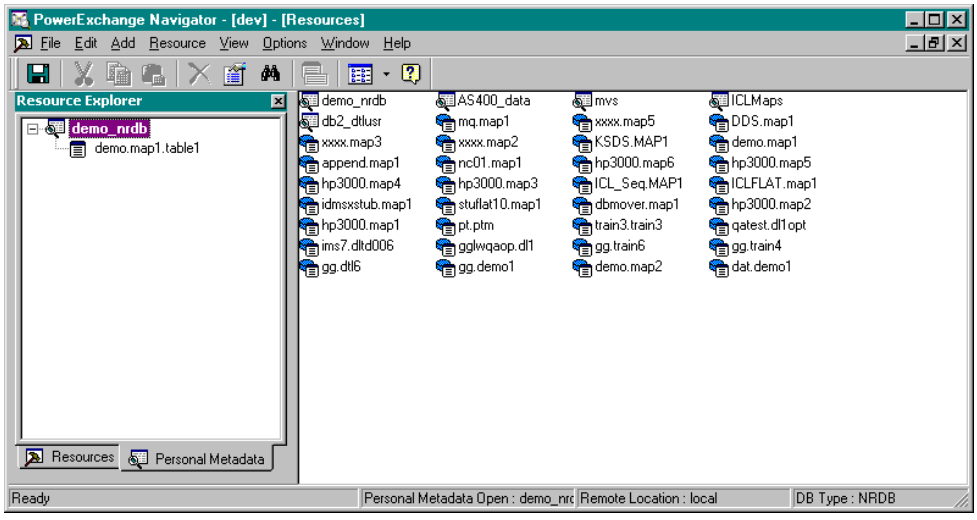
1. Enter a Schema of demo:



Note: Do not use an escape character to query database systems that are multibyte enabled.

2. Click **Finish**.

   The request runs. The results are visible in the left-hand pane:



3. If tables do not appear beneath the demo_nrdb metadata entry, the corresponding data maps might not have been exported. To export the tables, perform these steps:
   - On the Resources tab, double-click **Data Maps**.
   - Double-click an example map, for example demo.map2.
   - Click the table, in this case row_out.
   - Click the **Row Test** icon.
   - You are prompted to export the data map and accept the export. To run the Row Test, click **Go**.
   - When you return to the demo_nrdb personal metadata item, the exported data maps should be visible.

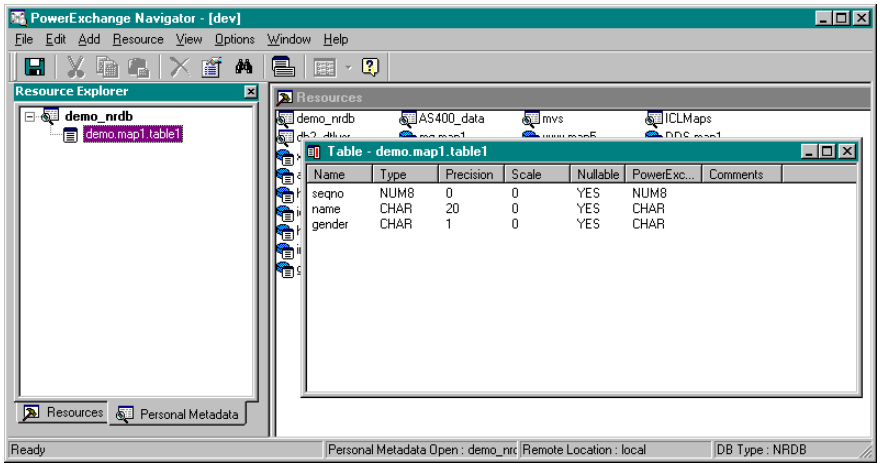
Step 3. Explore Metadata

Use the following procedure to explore metadata.

To explore metadata:

1. To explore metadata, place the cursor on one of the tables. Right-click the table.
2. Choose from one of the following options:
   - View properties for more information.
   - Use Explore to list the columns in the table.

   The following example shows the columns for table demo.map1. table1. By exploring metadata, you can see the column information for several tables at once:



Step 4. Perform Row Test

Warning: The Navigator allows you to select columns with datatypes that PowerExchange does not support. When you perform a database row test, be sure to exclude these columns. Otherwise, the row test fails.

To perform a row test:

1. Click the demo.map2.row_out table.
2. Click the **Row Test** icon.
3. In the Database Row Test dialog box, click **Go**.

   The row test data is displayed.

Row Tests and Non-standard Code Page Data

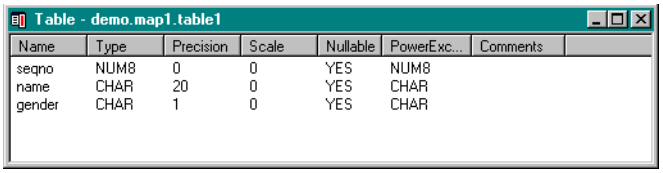
Non-standard code page data, including multibyte data, is available for selection from DB2 on both MVS and i5/OS, and the PowerExchange Navigator can display double-byte characters. The Navigator uses a UTF-8 code page enabling the display of a wide character set. To view the data in the correct format, ensure that you make changes to the PowerExchange configuration file on the platform from which you select the data. Review the following parameters:

- DB2CODEPAGE for DB2 on MVS
- DB2_BIN_CODEPAGE for DB2 on i5/OS

For more information about these parameters and about code page support, see the *PowerExchange Reference Manual*.

Metadata Column Display

The Metadata column display, which you can view by exploring the maps, offers several display options. You can sort the Name and Type columns by either alphabetical or reverse alphabetical order.



To sort the columns in alphabetical order, click the title box heading for either the Name or Type column.

To sort the columns in reverse alphabetical order, click the desired column heading again.

In the example, the Type column is sorted in reverse alphabetical order. A small arrow shape pointing up or down appears in the heading box to indicate the order in which the column is displayed.

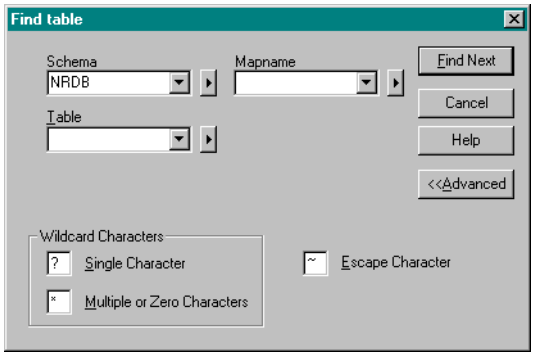
Search Facility for Metadata Display

The Search Facility described in "Searching in Data Maps" on page 102 can also be used with Personal Metadata to search for tables.

To use the search facility for metadata display:

1. From the menu bar, click **Edit > Find**.

   The following dialog box appears:



2. Perform a search by Schema, Mapname, or Table, depending on the database.

   You can specify the following wildcards:

   - An asterisk (*) represents one or more matching characters.
   - A question mark (?) represents one matching character.

CHAPTER 6

Registration Groups and Capture Registrations

This chapter includes the following topics:

Registration Group and Capture Registration Overview

For PowerExchange to capture changes, you must register source data sets and tables for change data capture using the PowerExchange Navigator.

When you create a registration group, PowerExchange automatically creates an extraction group and application group with the same name. When you define a capture registration for a source, PowerExchange automatically defines an extraction map for that registration. You can modify these default extraction maps and create new extraction maps based on the same capture registration.

After you create a capture registration, the Resource Inspector displays the properties of the capture registration.

The capture registration name is the name that you specify when you create the registration.

Note: The corresponding extraction map name, which PowerExchange automatically creates, has the following format: `dnninstance.regname_tablename`. For an explanation of the format of the extraction map name, see "Extraction Map Names" on page 128.

When you create a capture registration, PowerExchange assigns a capture registration tag to both the capture registration and the extraction map. The capture registration tag format varies depending on the source type. For more information, see "Capture Registration Tags" on page 114.

You can change only some capture registration properties. For more information, see "Editing Capture Registrations" on page 123.

Note: The maximum column length allowed for PowerExchange change data capture is 32,244 bytes.

Capture Registration Tags

When you create a capture registration, PowerExchange assigns a capture registration tag to both the capture registration and the extraction map. The capture registration tag format varies depending on the source type.

The following list describes the format of the capture registration tag for each source type:

Adabas

ADA*instance*DBID*nnnnn*FILEID*nnnnn*

In the capture registration tag format, the variables are as follows:

- *instance*. The collection identifier value specified when the registration group was created.
- DBID*nnnnn*. The Adabas DBID number.
- FILEID*nnnnn*. The Adabas FILEID number.

Datacom log-based

Datacom synchronous

DCM*instancereg_namen*

In the capture registration tag format, the variables are as follows:

- *instance*. The instance value specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- *n*. Version number.

DB2 for Linux, UNIX, and Windows

UDB*muf_namereg_namen*

In the capture registration tag format, the variables are as follows:

- *muf_name*. The MUF name value specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- *n*. Version number.

DB2 for z/OS

DB2*instancereg_namen*

In the capture registration tag format, the variables are as follows:

- *instance*. The database instance value specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- *n*. Version number.

i5/OS

AS4*instancereg_namen*

In the capture registration tag format, the variables are as follows:

- *instance*. The collection identifier value specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- *n*. Version number.

IDMS log-based

IDL*logsidreg_namen*

In the capture registration tag format, the variables are as follows:

- *logsid*. The Logsid value specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- *n*. Version number.

IMS log-based

```
IML recon_id reg_name 100000
```

In the capture registration tag format, the variables are as follows:

- *recon_id*. The recon identifier value specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- 100000. Static version number.

IMS synchronous

```
IMS.database_name.segment_name
```

In the capture registration tag format, the variables are as follows:

- *database_name*. The database name value specified when the data map was created.
- *segment_name*. The segment name value specified when the data map was created.

Microsoft SQL Server

```
MSS instance reg_name n
```

In the capture registration tag format, the variables are as follows:

- *instance*. The value PowerExchange generates from the database name value that was specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- *n*. Version number.

Oracle

```
ORA instance reg_name n
```

In the capture registration tag format, the variables are as follows:

- *instance*. The collection identifier value specified when the registration group was created.
- *reg_name*. The name assigned when the capture registration was created.
- *n*. Version number.

VSAM

The capture registration tag assigned to a VSAM data set depends on the length of the VSAM data set name in the data map.

If the VSAM data set name is less than or equal to 28 characters long, then the tag has the following format:

```
VSAM dataset_name
```

If the VSAM data set name is equal to or greater than 29 characters long, then the data set name in the tag is truncated to 23 characters long, and an *nnnnn* value is appended to the tag as follows:

```
VSAM dataset_name nnnnn
```

For example, a data set name of USERID1.VSAM.EDMSRCV.MAKEITA is 28 characters long. Consequently, it is assigned the following tag:

```
VSAMUSERID1.VSAM.EDMSRCV.MAKEITA
```

However, a data set name of USERID1.VSAM.EDMSRCV.MAKEITAB is 29 characters long and is assigned the following tag:

```
VSAMUSERID1.VSAM.EDMSRCV.MA00001
```

Subsequent data sets with names that are longer than 28 characters where the first 23 characters match USERID1.VSAM.EDMSRCV.MA are assigned a tag with an *nnnnn* value that is incremented by 1 as follows:

```
VSAMUSERID1.VSAM.EDMSRCV.MAnnnnn+1
```

For example:

```
VSAMUSERID1.VSAM.EDMSRCV.MA00002
```

Adding Registration Groups

Before you can add capture registration, you must create a registration group in the Data Capture folder of the Resource Explorer.

A registration group is a named group of individual capture registrations. It is a name of your choice. The related properties define the location of the PowerExchange Listener, the type of capture source, the name of the source subsystem and information such as user ID and password. All capture registrations within a registration group must access the same source type and instance of that source type.

Registration group information is stored locally on the Windows machine on which it was created and is unique for that machine. Accessing a PowerExchange Listener shows all registrations for the database instance associated with the registration group.

To add a registration group:

1. On the toolbar, click the **Add Registration Group** icon. Alternatively, right-click the Registration Group folder, and then click **Add Registration Group**.
2. The Add Registration Group dialog box appears. Enter values for the options in the dialog box.

   The number and names of some options vary based on the source type selected.

   The Add Registration Group dialog box has the following options:

   Name

   Specify the registration group name. The registration group name can be between 1 and 16 alphanumeric characters in length.

   Location

   Specify a valid location name that is specified the dbmover.cfg file on the Windows machine. The location indicates the node from which change data is captured. The location matches a node name in a NODE statement of dbmover.cfg or is set to local if the data source resides on the same Windows machine.

   For example, if you define a registration groups for DB2 for z/OS sources, the location should point to the PowerExchange Listener running on the same MVS image as the PowerExchange Agent, PowerExchange Logger, and DB2 ECCR.

   Type

   Specify a valid source type from one of the following:

   - **ADABAS.** Adabas files.
   - **AS4.** DB2 for i5/OS.
   - **DATACOM.** Datacom tables.
   - **DB2.** DB2 for z/OS tables.
   - **DB2UDB.** DB2 for Linux, UNIX, and Windows.
   - **IDMS_L.** IDMS tables registered for IDMS log-based change data capture.
   - **IMS.** IMS databases.
   - **MSSql.** Microsoft SQL Server tables.
   - **ORACLE**. Oracle tables.
   - **VSAM.** VSAM ESDS, KSDS, RRDS, and VRRDS data sets.

   UserID

   Optional. Specify a valid operating system or database user ID.

   The user ID is a valid database user ID and is required for DB2 for Linux, UNIX, and Windows, Microsoft SQL Server, and Oracle. This user ID must allow access to the specified location.

For MVS and i5/OS source types, the user ID is required if PowerExchange security is active. For more information about PowerExchange security, see the *PowerExchange Reference Manual*.

Password

Optional. Specify the password for the user ID specified in the **UserID** field.

Note: You can leave the **Password** blank for MVS and i5/OS source types. This prevents the password from being stored and causing potential password violations at a later date.

If you leave this field blank, the PowerExchange Navigator prompts for the password.

Collection Identifier

Specify a unique name for the group of registrations. A collection identifier name is also called an instance name.

This field appears when any of the following source types is selected:

- **ADABAS**
- AS4. For DB2 for i5/OS tables, specifies the instance name specified in the INST parameter of the AS4J CAPI_CONNECTION statement in the DBMOVER member of the dtllib/CFG file on the node specified in the Location option.
- ORACLE. For Oracle tables, specify the instance name specified in the collection_id parameter of the ORACLEID statement in the dbmover.cfg that is on the node specified in the Location option.
- **VSAM**

Database

For DB2 for Linux, UNIX, and Windows tables, specifies the database that contains the tables that are being registered.

Database Instance

For DB2 for z/OS tables, specifies the DB2 subsystem ID or the DB2 data-sharing group name that contains the tables. A database instance name is also called an instance name.

Database Name

For Microsoft SQL Server tables, specifies the database name of the database that contains the source tables.

Database Server

For Microsoft SQL Server tables, select the server where the database that contains the source tables resides.

Logsid

For IDMS tables used in IDMS log-based change data capture, specifies a unique name for the group of registrations. The name selected must be specified in the LOGSID parameter in the DBMOVER configuration parameters on the MVS system where the IDMS tables reside.

MUF Name

For Datacom tables, specifies the name of Multi-User Facility (MUF) in which changes for the tables occur.

Recon Identifier

For IMS synchronous change data capture, specifies the IMS subsystem ID in which changes for the tables occur.

For IMS log-based change data capture, specifies the name specified in the IMSID statement of the DBMOVER configuration file parameters on the MVS system where the RECON data sets reside.

Source Map Location

For IDMS tables used in IDMS log-based change data capture, specifies the PowerExchange location containing the data maps for the IDMS tables.

Add Registration

If selected, PowerExchange Navigator invokes the Add Capture Registration dialog so that a capture registration for the source type selected in the **Type** option can be added. By default, this option is selected. If cleared, only a registration group is added.

3. If **Add Registration** is selected, click **Next**. Otherwise, click **Finish**.

Adding Capture Registrations

PowerExchange uses capture registrations to determine which source data sets and tables participate in change data capture. Use the PowerExchange Navigator to register source data sets and tables for which you want to capture and extract changes. Capture registrations are stored in the CCT file at the source location specified in the capture registration group to which the capture registration belongs.

To add a capture registration:

1. When adding a new registration group, select **Add Registration** in the Add Registration Group dialog box.

   If the registration group exists, double-click the registration group name to open the group. Then, right-click the registration group name and click **Add Capture Registration**.

2. The Add Capture Registration - Name and Table Filter dialog box is displayed.

   Enter values for the options in the dialog box to filter the source tables displayed. The conditions are logically associated by the "and" connector.

   The Add Capture Registration - Name and Table Filter dialog box has the following options:

   Name

   Specify a name for the capture registration. A capture registration name can be up to eight alphanumeric characters in length and must begin with an alphabetic character. Use lowercase.

   Table Filter

   Optional. Specify filter criteria if you want to filter the list of tables that is retrieved from the specified location and instance. This feature can help you find the table that you want to select for registration. You can specify values in one or more the following fields, depending on your source type:

   Note: You can use the asterisk (*) wildcard to represent one or more characters at any point in a filter value.

   - **Creator/Schema** or **Schema** or **Owner**

     Specify the owner or schema name of the table.

   - **DBName/Definer** or **Mapname**

     Specify the source database name for DB2 tables or the IMS data map name for IMS databases.

   - **Table**

     Specify the table name.

   - **Respect Case**

     For source systems that have case-sensitive naming, such as DB2 and Microsoft SQL Server, select **Respect Case** if you want the case of your filter criteria to be honored during filtering. Otherwise, PowerExchange ignores case when searching for tables that match the filter criteria.

   Escape Character

   Optional. Specify the escape character that you want to use to delimit any character in **Table Filter** values that has a special meaning, such as %. In the filter value, place the escape character just before the character to delimit. For example, if the **Table** name is **my%tab** and the escape character is a forward slash (\), enter the table name as **my\%tab**.

3. Click **Next**.

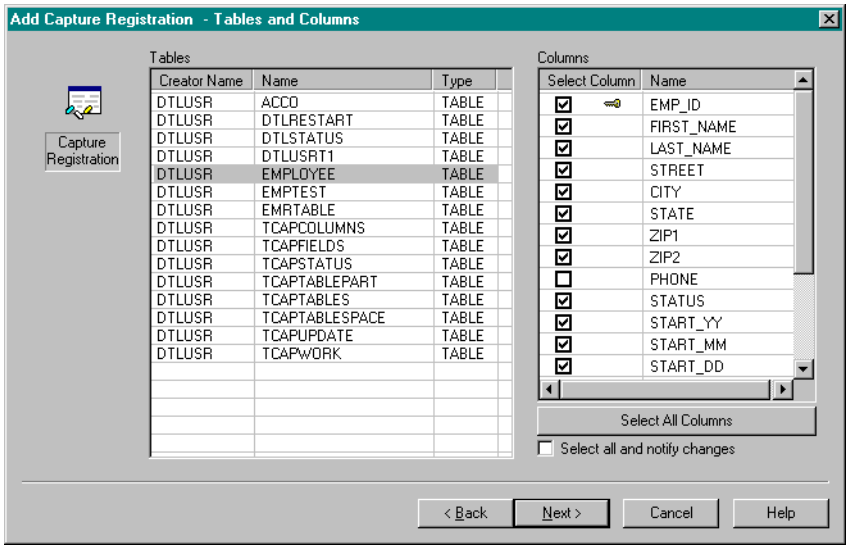
The Add Capture Registration - Tables and Columns dialog box appears. If you specified **Table Filter** criteria in the Add Capture Registration - Name and Table Filter dialog box, the **Tables** list is filtered based on your criteria.

4. Double-click a table in the **Tables** list. The columns in that table appear in the **Columns** list. Select the columns that you want to register. Then click **Next**.

   Columns marked with a key symbol are the index columns for that table. Key columns are used as part of the Condense processing. Be careful when deselecting key columns to ensure the "uniqueness" of the row is not compromised. Where a primary key is specified on the table, only those key columns are selected by default.

   Note: You can add only one capture registration for a source table at a time. Make sure that you select all of the columns for which to capture change data. An extraction process can extract data for a subset of those captured columns, but cannot extract data for other columns for which changes are not captured.

   The following example shows a list of DB2 for z/OS tables and the columns for the selected table:



The following table describes the columns in the **Tables** list in the Add Capture Registration - Tables and Columns dialog box:

Column	Description
Creator Name or Schema	Table creator
Name or Table	Table name
Type or DataSet Name	Object type such as TABLE, or data set name

The following table describes the columns in the **Columns** list in the Add Capture Registration - Tables and Columns dialog box:

Column	Description
Select Columns	Indicates whether a column is selected for registration.
Name	Column name
CCSID	For DB2 for Linux, UNIX, and Windows, the coded character set identifier (CCSID) related to the column
PWXCP	For DB2 for Linux, UNIX, and Windows, the PowerExchange internal representation of the code page
Codepage	For DB2 for Linux, UNIX, and Windows, the code page related to the column

To select columns for change data capture, use one of the following methods:

- Select the columns individually in the **Select Column** column. If you change a column that is not selected for capture, PowerExchange does not capture the change. PowerExchange only captures changes to columns registered for capture.
- Click **Select All Columns** to select all columns. If a change is made to any column within a row, that row is logged. This option is available only for source types that support selective column capture.
- Click **Select all and notify changes.** Removes the check boxes and selects all columns. Any change to a row causes that row to be logged.This option is available for DB2 and Oracle sources.

The key difference between **Select All Columns** and **Select all and notify changes** is seen when the source table changes. With specific column selection or **Select All Columns**, a column can be added to the source table with no impact. Columns that have been specifically selected and new columns added to the source table are ignored until they are selected. When you use **Select all and notify changes**, PowerExchange CDC fails if you change the table schema.

When you select **Select all and notify changes** for DB2 for z/OS tables and you change the table schema, the DB2 ECCR abends it reads the first change record for that table after the schema change. For more information about schema changes, see the *PowerExchange CDC Guide for z/OS*.

5. In the Add Capture Registration - Type dialog box, enter or select the following options:

Field	Description
Type	Displays the capture type for the data source, which can be one of the following values: - **Log-Based**. PowerExchange captures changes from the data source logs. - **Synchronous**. PowerExchange captures changes in a synchronous manner. This field is not editable.
Status	Select one of the following values: - **Inactive**. The capture registration is defined but not yet available for use for change capture. - **Active**. The capture registration is enabled for use for change capture upon the next start.
Condense	Type of condense processing to do for the capture registration: - **None**. Indicates that no condense processing is done. - **Part**. Indicates that PowerExchange Condense for MVS or i5/OS or the PowerExchange Logger for Linux, UNIX, and Windows performs “partial condense” processing. This processing stores all successfully committed changes for each source row in chronological order by end time. - **Full**. Indicates that PowerExchange Condense on MVS or i5/OS amalgamates all changes for each source row and stores only the first before image and last after image. Default is **None**. For more information about the Condense option, see “Defining the Condense Option” on page 121.
Database ID	For Adabas sources, displays the DBID of the table selected.
File Number	For Adabas sources, displays the file number of the table selected.
Supplement Log Group Name	For Oracle, the name of the supplemental log group that you want to create. PowerExchange generates DDL for creating supplemental log groups when you complete the capture registration. A supplemental log group must be defined for each table for which you want PowerExchange to capture Oracle change data. This supplemental log group must contain all table columns for which change data is to be captured.

Field	Description
Comment	An user-defined comment about the capture registration.
Execute DDL now	For Oracle, select this option if you want PowerExchange to execute the DDL that it generates for creating a supplemental log group when you complete the capture registration. Do not select this option if you want to execute the DDL later. A supplemental log group must be defined for each table for which you want PowerExchange to capture Oracle change data. This supplemental log group must contain all table columns for which change data is to be captured. You must also enter a **Supplement Log Group Name** value. **Note:** The GRANT statements in the ORACAPT.SQL file for the Oracle capture user do not include the authority that is required to run the DDL.

6. Click **Finish**.

   The PowerExchange Listener writes the capture registration to the CCT file on the source platform.

Defining the Condense Option

The Condense option defines the type of the condense processing PowerExchange should perform for the capture registration.

The following table describes the Condense options:

Condense Type	Description
Full	For MVS and i5/OS, specifies full condense processing. PowerExchange Condense amalgamates changes to deliver the first before image and last after image. For MVS, you can select the full condense option for tables or data maps that specify key columns. **Restriction:** PowerExchange does not support full condense processing for Adabas and IDMS log-based CDC sources. On i5/OS, you can select the full condense option for tables with primary keys or for DDS files defined with a unique key. **Note:** Select this option can only if the table has a unique key. PowerExchange Condense uses the unique key as part of the key in the condense file to maintain a single before and after image. If you update unique keys, specify KEY_CHANGE_ALW=Y in the PowerExchange Condense configuration file.
Part	Specifies partial condense processing. This type of processing provides all successfully committed changes to a row or record for a source in chronological order by end time.
None	The capture registration is not eligible for condense processing.

Adding a Capture Registration to an Existing Group

You can add capture registrations to an existing group.

To add a capture registration to an existing group:

1. Right-click the required Registration Group, and then click **Open**.
2. On the toolbar, click the Add Capture Registration icon. The Add Capture Registration - Name and Table Filter dialog box appears.
3. Complete the steps in “Adding Capture Registrations” on page 118.

Deleting a Registration Group

You can delete registration groups.

Deleting a Registration Group does not delete any Registration Entries. It only deletes the Group information that is held locally on the workstation.

To delete a registration group:

1. Select the appropriate Registration Group and click Delete.

   A confirmation dialog box displays.

2. Click Yes to delete.

   If no active registrations are outstanding for this group, the delete is accepted and the group is removed; otherwise, a further warning is issued stating the registration group has active registrations.

   The second warning is intended to prevent confusion when a group is deleted while an active registration remains that might still be capturing data.

Deleting a Registration Entry

You can delete registration entries.

To delete a registration entry:

- Select the appropriate Registration Entry and click the Delete button. If the registration is not active you are asked to confirm the deletion, otherwise you are prompted to make the registration history before deleting.

This action deletes the registration entry on the source platform. The next time the Collection Agent is restarted no more changes are captured for that registration.

Viewing a Registration Group

You can view registration groups.

To view a registration group:

- Select a group in the Resource Explorer. This displays the details for that group in the Resource Inspector. A complete list of registrations appears under the Registration Group in the Resource Explorer.

Viewing a Capture Registration

You can view the details of registration.

To view capture registration details:

- Double-click a Capture Definition Group from the Resource Explorer, Resource tab.

  Select a Capture Registration from the group list, and then double-click the Registration Name that you want to view.

Resource Explorer

The Resource Explorer displays a list of capture registration groups and the capture registrations within each group. To select and open a capture registration double click on the capture registration name. All the Registration details are displayed in the capture registration details dialog boxes.

Capture Registration Details

Capture registration details display the column information of the table referenced by the capture registration. This only displays details for those columns that are selected for capture.

This includes:

- Name
- Table Name
- Column No
- Type
- Length
- Scale
- Key

To make changes to an existing list of columns (for a given source) right-click on the registration name.

Resource Inspector

The Resource Inspector displays the registration properties of the capture registration. For example, Type is Synchronous and the Tag contents are used when writing changes to the PowerExchange Logger.

If you make any changes they must be confirmed by clicking on the Apply button. Once confirmed, the capture registration in the Resource Explorer changes color, which indicates that the source requires to be saved.

Editing Capture Registrations

After a data source has been registered for change capture, you might want to edit the registration to reflect any changes to your processing or underlying data.

You can change the following attributes of capture registrations:

- For DB2 and Microsoft SQL Server sources, you can add and remove columns. For more information, see “Adding or Removing Columns from a Capture Registration” on page 124.
- For all sources, you can change the status from inactive to active or from active to history. For more information, see “Changing the Status of a Capture Registration” on page 125.
- For all sources, you can change condense from none to part or part to none.

  For MVS and i5/OS sources, you can change condense for sources with keys from none or part to full.

  For more information, see “Defining the Condense Option” on page 121.
- For all sources, you can change the comments for the registration.

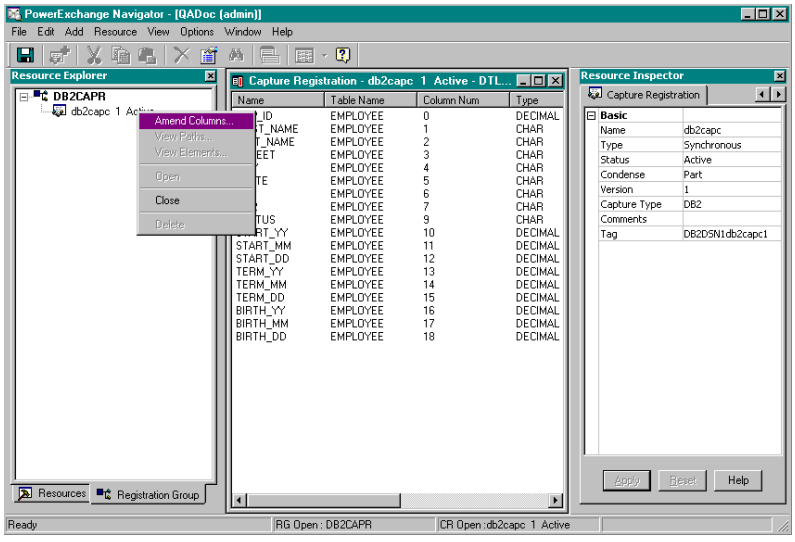
If you make changes to a capture registration, you must apply them by clicking **Apply**. After the changes are applied, the registration changes color indicating that you must save the registration.

To change any other properties of an existing registration, you must create a new capture registration and activate it.

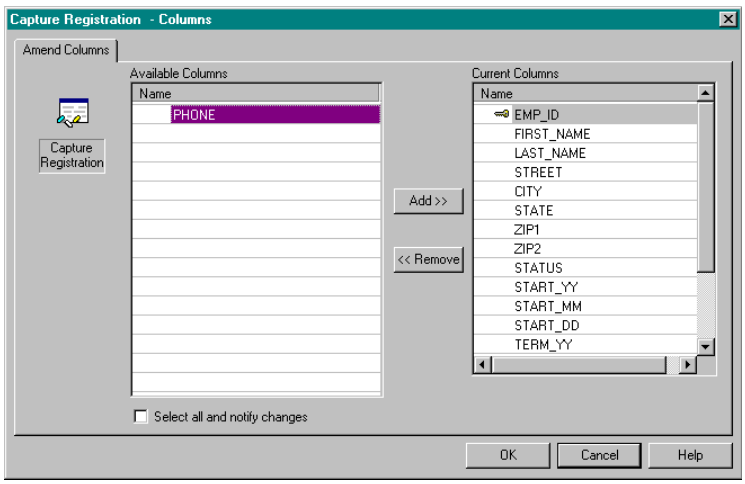
Adding or Removing Columns from a Capture Registration

To change a capture registration to add or remove columns:

1. Right-click on the capture registration, and then click **Open**.
2. Right-click on the capture registration, and then click **Amend Columns**. The following example shows a DB2 for z/OS capture registration:

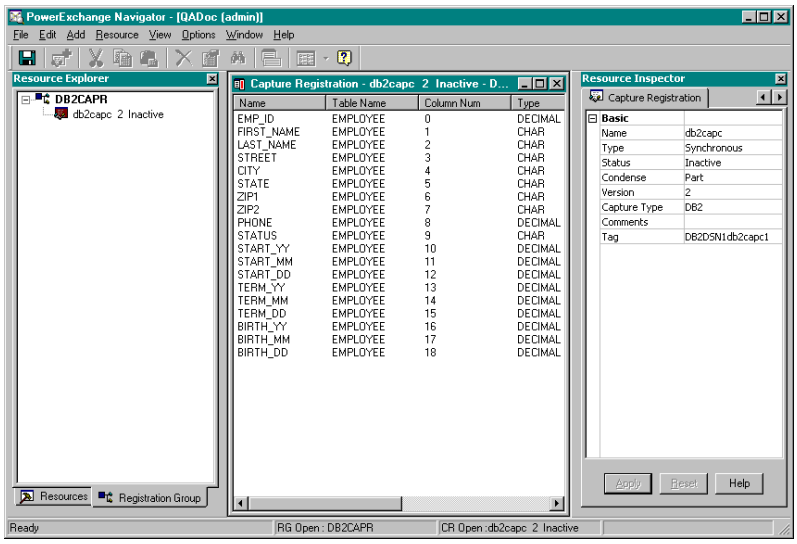


3. PowerExchange Navigator displays all available columns for the table.



4. Highlight the column(s) required, in this case PHONE, and click **Add**. Click **Remove** to remove columns.

5. When the change is accepted the Navigator displays the following information.



The window shows a red icon against the capture registration a version number of 2 against the registration. The PHONE column now appears in the capture registration in the middle pane of the window. Next time the registration group is opened, it shows the original Active capture as Version 1 and the new amended Version 2 as inactive.

The change is not effective until this version is set to Active as described previously. For more information, see “Setting the Status to Active” on page 125.

When this new version is set to Active, a confirmation dialog box appears.

After this change is applied, you can re-activate a registration that is set to history. After the registration is activated, PowerExchange captures the amended data. However, the original version is part of the original extraction definition. For more information about bringing the new definition into effect for extracting data, see “Changing the Capture Registration Associated with an Extraction Map” on page 135.

Changing the Status of a Capture Registration

You can change the status of a registration from inactive to active, or from active to history.

Setting the Status to Active

Creating the capture registration db2capc with a status of inactive prevents data from being captured until everything is in place for the data to be useful to apply to the target.

The capture registration usually remains inactive until the target has been materialized. When ready, the registration is set to Active.

To activate a capture registration:

1. Right-click the capture registration, and then click **Open**.
2. In the Resource Inspector, click **Status**.
3. Select **Active**. A small red square appears against the Status literal.
4. Click **Apply**.

   Note: The icon to the left of the registration turns red.
5. Right-click on the capture registration, and the click **Close**.
6. Click **Yes** in the confirmation dialog box, and the change is saved.

Setting the Status to History

If a capture registration is no longer required, you can disable change data capture for that source by setting the capture registration status to history. Open the Registration Group, and then double click or Open the Registration definition. In the following window, select the field that says Active, and then select History from the drop-down list.

When History is selected, a red square appears next to the amended field, which disappears when you click the Apply button.

When this has been carried out, the registration cannot be made active again. A new registration would need to be added to start capture again. If wishing to delete the registration setting to history has the advantage of retaining the capture information to date.

CHAPTER 7

Extraction Groups and Extraction Maps

This chapter includes the following topics:

Extraction Groups and Extraction Maps Overview

Use extraction maps to extract change data captured by PowerExchange. You can test change data extraction by using the database row test function in the PowerExchange Navigator.

When you create a registration group, PowerExchange creates an extraction group with the same name. When you create a capture registration, PowerExchange creates an extraction map with the same name and same columns. For information about extraction map names, see Extraction Map Names.

Additionally, when you create a capture registration, PowerExchange assigns a capture registration tag to both the capture registration and the extraction map. For more information about the format of the capture registration tag, see "Capture Registration Tags" on page 114.

You can alter the default extraction maps. You can also create additional extractions maps for the same capture registration by using the Add Extract Definition wizard.

Extraction Map Names

An extraction map name has the following format:

```
xnninstance.regname_tablename
```

The following table describes the format of the variables in the extraction map name:

Variable	Description
x	The first character defines the type of extraction map and is one of the following values: - d. Default extraction map. - u. User-defined extraction map.
nn	A one or two-digit value that defines the source type: - 1. DB2 for z/OS. - 2. IMS. - 3. DB2 for i5/OS. - 4. Adabas. - 5. IDMS synchronous change data capture. - 6. VSAM. - 7. Datacom. - 8. Oracle. - 10. Microsoft SQL Server. - B. DB2 for Linux, UNIX, and Windows. - D. IDMS log-based change data capture.
instance	The instance value specified when the registration group was created. The instance value varies by source type. For more information, see “Adding Registration Groups” on page 116.
regname	The name specified when the capture registration was created.
tablename	The table name of the relational source table or from the source data map.

For extraction maps, the variables *xninstance* are called schema, and *regname_tablename* are called map name. When you open the extraction group, PowerExchange Navigator displays the extraction maps as *schema.regname*. When you extract changes using an extraction map, specify the full extraction map name of *schema.map_name.*

For example, when using database row test, PowerExchange Navigator generates the following SQL statement to extract the changes:

```
SELECT * FROM schema.regname_tablename
```

You can select all columns or specific columns.

Note: You can override the schema name and map name by using the **Schema Name Override** and **Map Name Override** fields in the PowerExchange Client for PowerCenter (PWXPC).

Adding Extraction Groups

Extraction maps are defined under the Extraction Group in the Resource Explorer. By default, whenever a registration group is defined an extraction group with the same name is created. There is normally no need to define extraction groups manually, but you can create extraction groups.

To add an extraction group:

1. On the toolbar, click the **Add Extraction Group** icon. Alternatively, right-click **Extraction Group**, and then click **Add Extraction Group**.
2. In the Add Extraction Group dialog box, enter or select the following information:

Field	Description
Name	The name of the extraction group.
Location	The name, as defined in the dbmover.cfg file, where the PowerExchange Listener is located.
Type	The source type of data to be extracted.
UserID/Password	User ID and password to identify the user and check their authorizations.
Collection Identifier	Matches the collection ID value that you specified for the associated registration group. For Oracle, it must also match the ORACOLL parameter in the ORCL CAPI_CONNECTION statement of dbmover.cfg. This field appears for the following source types: - Adabas - DB2 for i5/OS - Oracle - VSAM
Database	For DB2 for Linux, UNIX, and Windows sources only, specifies the name of the database that contains the source tables. Specify the same value as specified for the matching registration group.
Database Instance	For DB2 for z/OS sources only, the DB2 subsystem or group name. Specify the same value as specified for the matching registration group.
Database Server and Database Name	For Microsoft SQL Server sources only, the name of the database server and the name of the database on that server that contains the source tables. Specify the same values as specified for the matching registration group.
Logsid	For IDMS log-based sources only, a unique name for the group of registrations. Specify the same value as specified for the matching registration group.
MUF Name	For Datacom sources only, the name of Multi-User Facility (MUF) in which changes occur. Specify the same value as specified for the matching registration group.
Recon ID	For IMS synchronous sources only, the IMS subsystem ID in which changes for the tables occur. For IMS log-based sources, the name specified in the IMSID statement of the DBMOVER configuration file parameters on the MVS system where the RECON data sets reside. Specify the same value as specified for the matching registration group.
Source Map Location	For IDMS log-based sources only, the PowerExchange location containing the data maps for the IDMS tables. Specify the same value as specified for the matching registration group.
Add Extraction Definition	Select this check box if you want to define extraction map details. Clear this check box if you want to define only the extraction group and define the extraction map details later.

3. Click **Next** or click **Finish**.

   All existing extraction maps for the specified type, database instance, and location are displayed.

Adding Extraction Maps

To add an extraction map:

1. Right-click an extraction group and click **Add Extract Definition**.
2. In the Add Extract Definition dialog box, you can view the schema name and enter a map name and table name as follows:

Field	Description
Schema Name	Read-only. A name for the schema portion of the extraction map name based on the following naming convention: `unninstance` Where: - u. Indicates that the extraction map is user-defined. - *nn*. A one- or two-digit value that indicates the source type, for example, `10` represents Microsoft SQL Server. For more information about the source type portion of the schema name, see "Extraction Map Names" on page 128. - *instance*. The database instance name. The schema name is unique within a specific PowerExchange Listener, database instance, and database type.
Map Name	A name that is specific to this extraction map. Default is the capture registration name. Maximum length is eight characters.
Table Name	A name for the table portion of the extraction map name. Default is the actual table name. Maximum length is 20 characters.

The extraction map name has the following format:

```
SchemaName.MapName_TableName
```

In this example, the extraction map name is:

```
u1dsn1.DB2MAP.DB2TAB
```

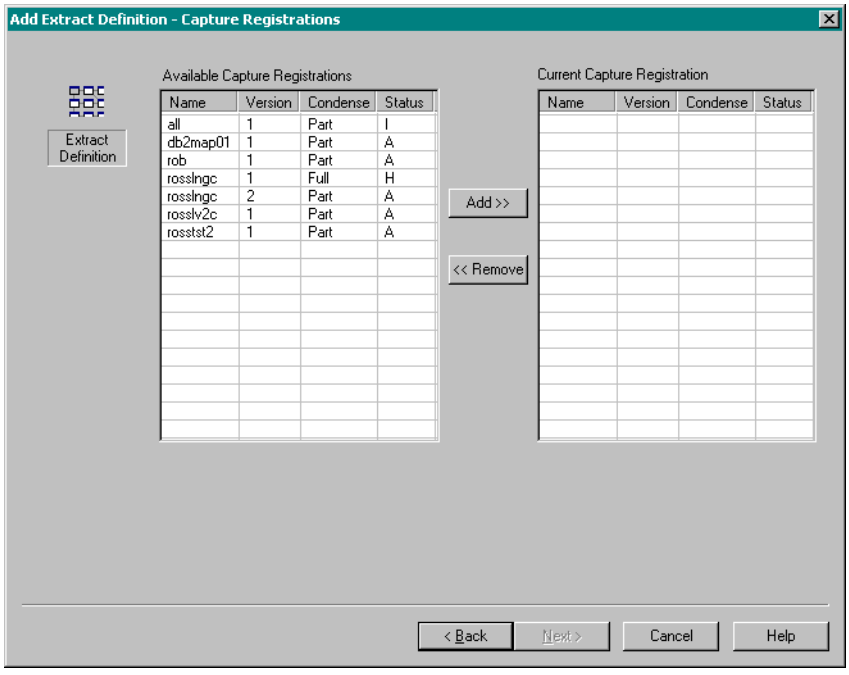
Adding an Extraction Map - Capture Registration Dialog Box

Using this dialog box, the user can add a version of a capture registration to the extraction process definition.

The Available Capture Registrations list displays the Capture Registrations that are available for selection. The Current Capture Registrations list displays the name of the Capture Registrations that have been selected for the extraction map.

To add a version of a capture registration to an extraction map:

- Highlight the required capture registration, and then click **Add**.



The following table describes the columns in the **Available Capture Registrations** list in the Add Extract Definition - Capture Registrations dialog box:

Field	Description
Name	Displays the name of the capture registrations that are available for selection.
Version	Displays the version number of the capture registration.
Condense	Displays the condense option (**None**, **Part** or **Full**) that is selected for the capture registration.
Status	Displays one of the following status values for the capture registration: - **I**. Indicates inactive. The registration is defined but is not available to be used for change data capture. - **A**. Indicates active. The registration is available for use for change data capture.

The following table describes the columns in the **Current Capture Registration** list in the Add Extract Definition - Capture Registrations dialog box:

Field	Description
Name	Displays the name of the capture registration that has been selected.
Version	Displays the version number of the capture registration.
Condense	Displays the condense option (None, Part or Full) that is selected for the capture registration.
Status	Displays the status of the capture registration.
Add button	Adds the capture registration selected in **Available Capture Registrations** list to the **Current Capture Registrations** list.
Remove button	Removes the capture registration in **Current Capture Registration** list.

Specifying Extraction Criteria

A variety of values can be specified when running an extraction process. When using either PWXPC through PowerCenter or the PowerExchange ODBC drivers, these criteria have to be defined as part of the extraction process.

For more information about using PWXPC to extract changes, see *PowerExchange Interfaces for PowerCenter*.

PowerExchange ODBC drivers allows various overrides using escape sequences:

- `DTLXTYPE=`
- `DTLAPP=`
- `DTLDSN=`

Type	Parameters	Description
SL	None	Extract all new captured data since the last extraction.
RS	-n	Default DTLXTYPE Rerun the last specified extraction process, which is either the last extraction that was run or the one that is specified with the Reset Start Point Wizard in the Application dialog box. For more information about Application Groups, see "Application Groups and Capture Application Overview" on page 151.

For more information, see the *PowerExchange Reference Manual*.

Adding an Extraction Map to an Existing Extraction Group

Use the following procedure to add an extraction map.

To add an extraction map to an existing group:

1. Right-click the extraction group and click **Open**.
2. On the toolbar, click the **Add Extract Definition** icon.

   The Add Extract Definition - Name dialog box appears. For more information about this dialog box, see "Adding Extraction Maps" on page 130.
3. Follow Steps 2 and 3 in "Adding Extraction Groups" on page 128.

Deleting an Extraction Group

Use the following procedure to delete an extraction group.

To delete an extraction group:

1. Select the extraction group and click **Delete**. A confirmation dialog box opens.
2. Click **Yes**. Deleting the extraction group does not delete the extraction maps. Deleting extraction groups only removes the group from the Windows machine on which the delete is done.

Deleting an Extraction Map

Use the following procedure to delete an extraction map.

To delete an extraction map:

- Select the extraction map and click **Delete**. A confirmation dialog box displays.

  After the entry is deleted, it is no longer possible to extract any changed data by using that extraction map. Verify that corresponding changes are made to any PowerCenter mappings and sessions.

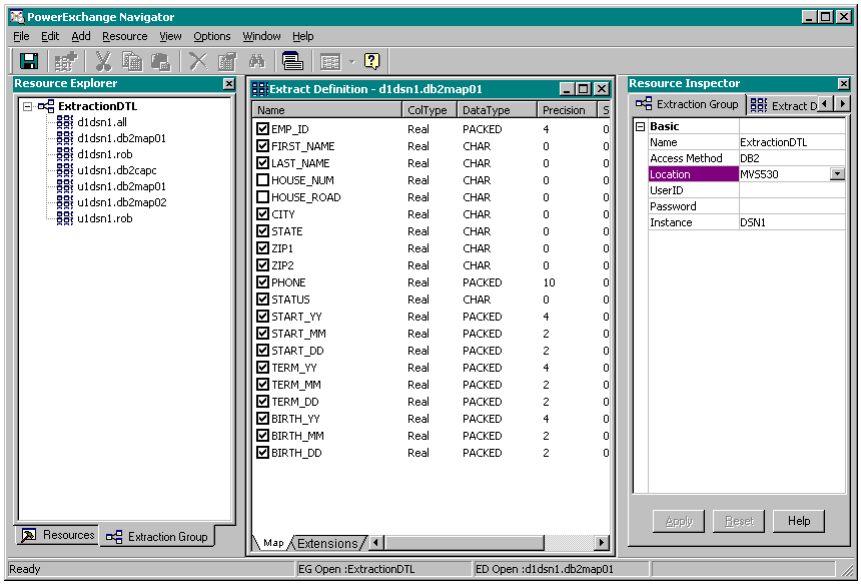
Viewing an Extraction Map

Use the following procedure to view an extraction map.

To view an extraction map:

1. Double-click the extraction group in the Resource Explorer.
2. Select an extraction map, and then double-click it to open it.

   The following example shows an extraction map for a DB2 for z/OS source:



The following appears:

- **Resource Explorer.** Displays a list of extraction groups and the extraction maps within each group. To select and open an extraction map, double-click on the Extraction Name. All the Extraction details are displayed in the Extract Definition Details dialog boxes.
- **Extract Definition.** It is possible to remove columns from the extraction map by de-selecting them by un-checking the associated box. As a default the automatically generated columns are not displayed. Right click to display these columns. They can be selected as part of the extraction map. Select the File, Save menu option to save any changes.
- **Extraction Details.** Displays the Name, Version and referenced Table of the extraction map.
- **Table Details.** Displays the column information of the table referenced by the extraction map, including: name, type, precision, scale, and length.

You can right-click in the Extract Definition window, and then select **Show Auto Generated Columns**. The following internal columns display:

Column	Description	Data Type	Length
DTL__CAPXRESTART1	A value that represents the position of a change record in the change stream. The value of this column is always strictly ascending.	VARBIN	255
DTL__CAPXRESTART2	A value that represents the restart token that PowerExchange uses to determine the point from which to start reading change data from the change stream if the extraction were to end at this point in time. For changes in full condense files, the value is the instance name from the registration group of the capture registration.	VARBIN	255
DTL__CAPXUOW	A value that represents the position in the change stream of the start of the unit-of-work for the change record.	VARBIN	255
DTL__CAPXUSER	The user ID making the change, as recorded by the data source software. The contents of this field may differ for the following data sources: - For DB2 for i5/OS data sources, the value is the user ID of the user that made the change. If you specify LIBASUSER=Y on the AS4J CAPI_CONNECTION statement, the value is the library and file name to which the change was made. - For DB2 for z/OS data sources, the value depends on the value of UIDFMT parameter on the LRAP CAPI_CONNECTION. If you do not specify UIDFMT, then the field contains the user ID that performed the operation. For more information about UIDFMT, see the *PowerExchange CDC Guide for z/OS*. - For Microsoft SQL Server, the value is always null. Microsoft SQL Server does not record this information. - For Oracle, the value may be either the user ID of the user who made the change or null. Oracle LogMiner provides the user ID if it is known.	VARCHAR	255
DTL__CAPXTIMESTAMP	Timestamp for when the change was made to the data source, as recorded by the source DBMS. The timestamp format is: `YYYYMMDDHHMMSSnnnnnn` Where: - *YYYYMMDD* is the date in year-month-day format - *HHMMSSnnnnnn* is the time in hours(HH), minutes (MM), seconds (SS), and microseconds (nnnnnn) Note: Oracle does not support microseconds in the timestamp.	CHAR	20

Column	Description	Data Type	Length
DTL__CAPXACTION	A single character that indicates the type of change operation: - I for INSERTs - D for DELETEs - U for UPDATEs	CHAR	1
DTL__CAPXCASDELIND	For DB2 for z/OS sources only, a character indicator for delete operations that DB2 generates for tables that specify the ON DELETE CASCADE clause: - Y for dependent rows that DB2 deletes as a result of a cascade delete - N for all other delete operations	CHAR	1
DTL__BI	For UPDATE operations, the value of the before image of the selected column in the change record. For more information about before image columns, see “Using Change Indicator and Before Image Columns” on page 137.	Datatype and length of the source column.	
DTL__CI	For UPDATE operations, a Y or N value that indicates whether the selected column was changed. For INSERT and DELETE operations, a null value. For more information about change indicator columns, see “Using Change Indicator and Before Image Columns” on page 137.	CHAR	1

In addition, null indicator hidden columns are shown adjacent to nullable columns and a length indicator when these are of variable length.

- **Resource Inspector.** The Extraction Group tab displays the connection properties of the extraction map. If you make any changes, click **Apply** to apply the changes.

Removing Columns from the Extraction Map

If a field is no longer required by the processing application but it is still present in the source database, the least intrusive change is to remove the field from the extraction map. The view of the underlying data is determined by the capture registration. You can remove columns from the extraction map. PowerExchange capture changes for sources using the capture registration. PowerExchange extracts the captured changes using the extraction map. You do not need to include all captured columns in an extraction map.

After the extraction map has been changed, subsequent extraction processing does not pick up any data from columns that have been deselected.

To remove to columns from an extraction map:

1. Click the extraction map to highlight it.
2. Right-click the extraction map, and then click **Open**.
3. In the Extract Definition window, clear the box beside the column to be removed.

This process does not prevent the data being captured. Selecting the column again at a later date makes that data available again for extraction through this extraction map.

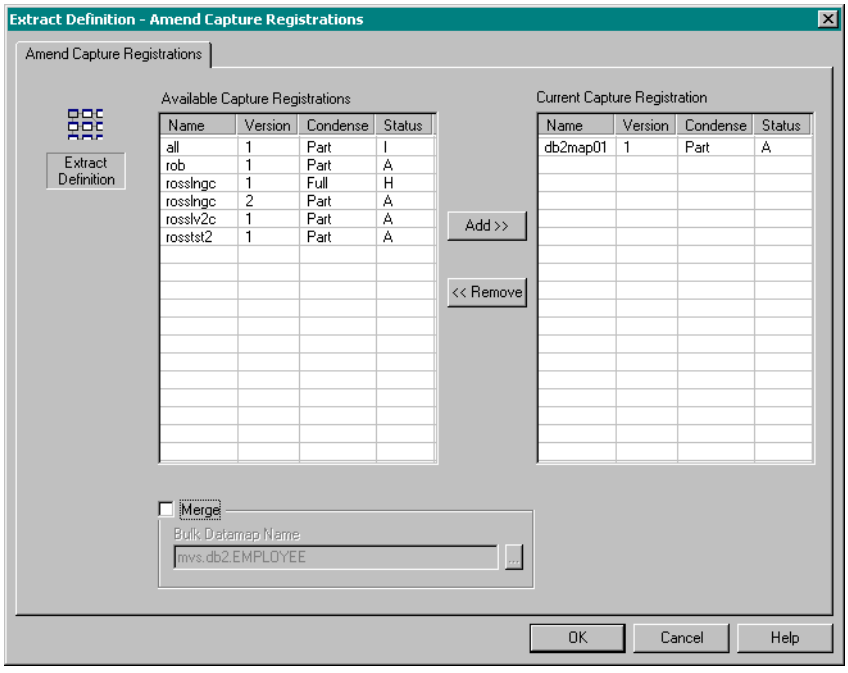
Changing the Capture Registration Associated with an Extraction Map

Use the following procedure to change the capture registration associated with an extraction map.

To change the capture registration associated with an extraction map:

1. Click the extraction map to highlight it.
2. Right-click the extraction map, and then click **Open**.
3. Right-click the extraction map, and then select **Amend Capture Registrations** to select a different capture registration.

   For information on **Amend Change Indicator/Before Image Extensions**, see "Using Change Indicator and Before Image Columns" on page 137.
4. Use **Amend Capture Registrations** if a new version of the capture registration has been created. The following shows how a change the extraction map to use version 2 of the capture registration. Only a single Capture registration can be selected.



The following table describes the columns in the **Available Capture Registrations** list:

Column	Description
Name	Displays the name of the Capture Registrations that are available for selection.
Version	Displays the version number of the Capture Registration.
Condense	Displays the condense state of the Capture Registration: None, Part or Full.
Status	Displays the status of the Capture Registration. This can have one of two values: - I (Inactive). The registration is to be defined but is not to be used by the Collection Agent. - A (Active). The registration is to be defined and used by the Collection Agent.

The following table describes the columns in the **Current Capture Registration** list:

Column	Description
Name	Displays the name of the Capture Registration that has been selected.
Version	Displays the version number of the Capture Registration.
Condense	Displays the condense state of the Capture Registration: None, Part or Full.

Column	Description
Status	Displays the status of the Capture Registration.
Add	Adds the Capture Registration selected in Available Capture Registrations list to the Current Capture Registrations list.
Remove	Removes the Capture Registration selected in Current Capture Registrations list.

Now the version 1 capture registration is removed and the version 2 registration added.

5. Click **OK** to save the changed extraction map.

   The resulting window shows the new version in the middle pane.

Version Indicator in Extraction Maps

The version indicator is used to identify the current version of the extraction map. The extraction map is represented as a data map. The extraction process uses this data map to read and format the data. The qualifier used in the SQL request used by the application to retrieve the changes specifies a data map name, such as the extraction map name.

What happens if the registration changes when a column is added or dropped? The layout of the extracted data changes. If the application retrieves the changes using the old data map, the data might be affected, the results are unpredictable. When a column is added it might just work but this must not be relied upon.

The requesting application is not aware of the version identifier. The way to correct that incompatibility is by using the extraction map process and adjusting the correct version identifier to the extraction map. A wizard guides you through that process.

What other changes are required if the version identifier changes? By making the changes in the extraction process only a part of the changes are completed. If the application is an ETL product the 'SOURCE' definition has to be adjusted so that the new columns are recognized or old ones are being dropped. The 'TARGET' definitions have to be made aware of the changes in the data layout.

By changing the version identifier a manual server process ensures that there is no more captured data that has not been extracted. Using the version identifier, it is always possible to extract captured data with the correct data map by defining an extraction process that specifies the specific version for a given capture file.

Before a registration is changed, all the data for that registration must first have been extracted. The new registration can then be given its new version number and its data can then be extracted.

When defining a new capture registration, PowerExchange automatically defines a corresponding extraction map with a version identifier of 1. All further changes to that extraction map must be made manually.

Using Change Indicator and Before Image Columns

Change indicators enable you to determine if a particular field in the registered table has changed.

Use it, for example, in an SQL WHERE clause to retrieve only those columns that have changed rather than the entire table.

To achieve this level of functionality the extraction data map can be modified to add Change Indicator (CI) and Before Image (BI) columns.

For the After Image (AI), a change indicator (CI) can be added to each column in which you are interested. For Updates this indicates, on retrieval, whether (Y) or not (N) the column has been changed. Inserts and Deletes always return a null.

By adding the Before Image (BI) column, the Before Image of the changed field is also retrieved in addition to the normal retrieval of the After Image.

Note: The change indicator field is only set to 'Y' if the update causes the column value to change.

Selecting CI and BI Columns in an Extraction Map

You must use the PowerExchange Navigator to edit the required extraction map. In the following example, the Change Indicator and Before Image columns are added to the CITY field.

To select CI and BI columns in an extraction map:

1. In the PowerExchange Navigator, open the extraction map.
2. Position the cursor in the Extract Definition pane.
3. Right-click the extraction map, and then click **Amend Change Indicator/Before Image Extensions**.

   The Extract Definition - Amend Columns dialog box appears. The Change Indicators tab is active.
4. From the Available Columns list, select the columns that you want to amend (for example, the CITY column) and click **Add**.

   You can also select the CITY column. Right-click and select **Add**.

   The CITY column is moved to the Selected Columns list.
5. Click the Before Images tab.
6. From the Available Columns list, select the columns that you want to amend. For example, select the CITY column and click **Add**.

   You can also select the CITY column. Right-click and select **Add**. The CITY column is moved to the Selected Columns list.
7. Click **OK**.

   Note: The new DTL__CI_CITY and DTL_BI_CITY columns are added to the list. The column type of these new columns is CI and BI respectively and the Real Column field displays the source CITY column.The CITY column displays DTL__CI_CITY in the CI Column field and DTL__BI_CITY in the BI Column field.
8. To display full details of the new fields, click the Extensions tab.

Testing the Column Changes Using Database Row Test

Use the PowerExchange Navigator Row Test facility to test the changes that you have made to the extraction map.

In the following example, the record of employee id 3 has been updated to show a new CITY of Shanghai.

To test column changes with row test:

1. In the PowerExchange Navigator, select the extraction map that you require to test.
2. Select **File > Database Row Test**.
3. Ensure that the correct information is entered into the Row Test dialog box.

Field	Description
Location	The location for the target node as set in the dbmover.cfg file.
UserID/Password	The user ID and password used to access the data.
Application Name	Required for CAPX/RT and is the application that was created in the PowerExchange repository (CCT). This can be created using DTLUAPPL.
SQL Statement	The SQL statement is automatically generated.

4. Click **Go**.

   The results of the data retrieval are displayed. These represent the changes that have occurred on the table that you registered for data capture.

   Note: The new Control Indicator (CI) /Before Image (BI) columns are visible in the Row Test results show that:

   - The column was changed (`DTL_CI_CITY=Y`).
   - The After Image of the record (`CITY=Shanghai`).
   - The Before Image of the record (`DTL_BI_CITY=innercity`).

CHAPTER 8

Using Database Row Test

This chapter includes the following topics:

Database Row Test Overview

You can use the database row test to preview data for all data sources and to generate restart tokens for PowerExchange change data capture sources.

Database row test can be used for objects in the following folders in PowerExchange Navigator:

- **Extraction Groups in the Data Capture folder.** You can use database row test with an extraction map to preview captured change data.
- **Data Maps folder.** You can use database row test with data maps validate data maps and to preview source data.
- **Personal Metadata folder.** You can use database row test to preview source data for tables displayed in personal metadata profiles.

You do not need to select an object to use database row test. You can also use database row test in the Resource Explorer window.

Performing a Database Row Test

Use the following procedure to perform a database row test of data maps, extraction maps, or personal metadata profiles. The test accesses columns in a data source and displays them in tabular format. The results of a data map or extraction map row test indicate that PowerExchange can retrieve data when bulk data movement or change data extraction runs.

Note: You can also issue a PowerExchange Listener LISTTASK or STOPTASK command in a database row test. For more information, see "Issuing a PowerExchange Listener Command in a Database Row Test" on page 142.

To perform a row test:

Note: The procedure for testing an extraction map or personal metadata profile is similar to the following procedure.

1. Open a data map.
2. On the Data Map tab, select the table view.

   Note: To retrieve data for specific columns only, select these columns in the Table window. PowerExchange adds the columns that you select to the SELECT statement in the Database Row Test dialog box.
3. Click **File > Database Row Test**.
4. In the message box that prompts you to send the data map to a remote location, click **Yes**.
5. In the Data Map Remote Node dialog box, enter the remote node from which data is extracted in the **Location** list.
6. Click **OK**.
7. In the Database Row Test dialog box, select a **DB Type** value. This value determines which fields are available for input in the Database Row Test dialog box.

   Note: Database row tests that use the CAPX DB type read changes from condense files created by PowerExchange Condense. Database row tests that use the CAPXRT DB type read changes from the change stream.
8. In the **Fetch** list, select one of the following options:
   - **Columns.** Metadata request to retrieve information about columns.
   - **Data.** Retrieve data rows.
   - **Foreign keys.** Metadata request to retrieve information about Foreign keys.
   - **Primary keys.** Metadata request to retrieve information about primary keys.
   - **Records.** Metadata request to retrieve information about NRDB and NRDB2 records.
   - **Schemas.** Metadata request to retrieve schema names for the DB_Type specified.
   - **Tables.** Metadata request to retrieve information about tables.

   To preview data, select **Data**.
9. In the SQL Statement box, you can modify the SQL statement generated by PowerExchange. By default, PowerExchange generates the following SELECT statement for data maps:

   ```
   select * from table
   ```

   For more information about PowerExchange nonrelational SQL, see the *PowerExchange Reference Manual*.
10. Click **Go**.

    Results appear in the Database Row Test Output window at the bottom of the screen. By default, PowerExchange displays ten rows of data. You can display up to 999 rows of data by changing the value in the **Get Rows** list.

Issuing a PowerExchange Listener Command in a Database Row Test

To issue a PowerExchange Listener LISTTASK or STOPTASK command in a database row test, perform the following steps.

To issue PowerExchange Listener commands in a database row test:

1. Open a data map.
2. On the Data Map tab, select the table view.
3. Click **File > Database Row Test**.
4. In the message box that prompts you to send the data map to a remote location, click **Yes**.

5. In the Data Map Remote Node dialog box, select the remote node from which data is extracted in the **Location** list.
6. Click **OK**.
7. In the Database Row Test dialog box, click **TASK_CNTL** in the **DB Type** list.
8. Optionally, enter an application name in the **Application** box for the STOPTASK command. The application name is the name for the active extraction process that you want to stop. This name is included in the LISTTASK command output.
9. Click **Go**.

Specifying Access Methods

If Tables or Columns is selected for NRDB, then the Access Methods list is enabled. It is possible to specify a list of valid access methods and the row test result filters the result set for the access methods specified.

The Get Rows box allows you to specify how many rows to acquire on this next request. The default is 10, but you can it to a maximum of 999. Extracting large amounts of data and adding it to these Windows tabular displays is not rapid. It is recommended that only a manageable number of rows be retrieved in a single request.

PowerExchange SQL for Metadata Requests

In previous examples, only data was accessed from the source. However, you can also retrieve metadata to describe the tables and columns. There is a special PowerExchange syntax to do this, DTLDESCRIBE, which is explained fully in the chapter on nonrelational SQL Syntax in the *PowerExchange Reference Manual*.

Note: The DTLDESCRIBE syntax is valid for both relational and nonrelational sources.

For the metadata requests, the default value for the Schema name i s converted to the case that is standard for the source database, such as uppercase for Oracle. For example, if you enter *scott* or *SCOTT* in the Schema field, you get the same results, because *scott* is converted to uppercase. By checking the Respect Case box, defaulting to the database case is not done. Hence a Schema name of scott produces no results, whereas SCOTT is successful.

The metadata qualifiers support the following wildcards:

- An asterisk (*) represents one or more matching characters.
- A question mark (?) represents one matching character.

These characters were chosen, as they are not likely to be used in table and column names. If they are being used, precede them with the Escape Character ~. For example a request for tab* would list all tables beginning with tab, whereas a request for tab~* would only list the table that was named tab*.

Specifying Advanced Parameters in a Database Row Test

If you select CAPX, CAPXRT, DB2390IMG, or EMR in the **DB Type** list in the Database Row Test dialog box, click **Advanced** to display the Advanced Parameters dialog box.

In the Advanced Parameters dialog box, enter or select parameters for the database row test that override settings in the PowerExchange configuration file.

The fields that appear on the Advanced Parameters dialog box differ depending on the value you select in the **DB Type** list.

CAPX Advanced Parameters

In the CAPX Advanced Parameters dialog box, enter or select the following information:

Parameter	Values	Description
Extract Type	SL,RS	- SL. Extract all data since last extract run. - RS. Restart previous, or specified, extract.
Image Type	BA,AI,TU	- BA. Before and after images are delivered as Delete and Insert records for any change. - AI. Captures only the data after image (the latest change). - TU. Before and After images are delivered with an action indicator of change, rather than Delete and Insert records as delivered by BA. The application that processes the row must be able to apply changes rather than just delete and insert affected rows.
Timeout	0 to 86400	Sets the maximum approximate time in seconds to wait for data on a queue before returning EOF. The following values indicate the following actions: - 0 indicates that EOF is returned immediately that there is no more data. - 86400 indicates that EOF is never returned and the job waits forever.
No Progress Update		Default is selected. If No Progress Update is selected, PowerExchange does not treat this extract as a true application extract.
AS400 Instance		Overrides the instance name of the AS400 database.
Extraction Schema		Overrides the schema specified in the extraction map.

CAPXRT Advanced Parameters

In the CAPXRT Advanced Parameters dialog box, enter or select the following information:

Parameter	Values	Description
Extract Type	SL,RS	- SL. Extract all data since last extract run. - RS. Restart previous, or specified, extract.
Image Type	BA,AI,TU	- BA. Before and after images are delivered as Delete and Insert records for any change. - AI. Captures only the data after image (the latest change). - TU. Before and After images are delivered with an action indicator of change, rather than Delete and Insert records as delivered by BA. The application that processes the row must be able to apply changes rather than just delete and insert affected rows.
Timeout	0 to 86400	Sets the maximum approximate time in seconds to wait for data on a queue before returning EOF. The following values indicate the following actions: - 0 indicates that EOF is returned immediately that there is no more data. - 86400 indicates that EOF is never returned and the job waits forever.
No Progress Update		Default is selected. If No Progress Update is selected, PowerExchange does not treat this extract as a true application extract.
AS400 Library/Journal		Specifies the fully qualified library and journal name to use instead of that specified in the PowerExchange change capture registration. For example: `STQA/NEWJOURNAL`

Parameter	Values	Description
Oracle Instance		Overrides the Oracle instance information for a given Oracle Collection Id enabling you to use a single set of registrations to capture data from multiple Oracle instances. This overrides the second value of the ORACLEID statement in the PowerExchange configuration file. For example: `ORACLEID=(coll_id,oracle_sid,connect_string, cap_connect_string)` Used in conjunction with the Oracle Connection string. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture determines the value of the other keyword by using the PowerExchange configuration file.
Oracle Connection String		Overrides the Oracle connection information for a given Oracle Collection Id enabling you to use a single set of registrations to capture data from multiple Oracle instances. This value overrides the fourth value of the ORACLEID statement in the PowerExchange configuration file. For example: `ORACLEID=(coll_id,oracle_sid,connect_string, cap_connect_string)` Used in conjunction with the Oracle Instance. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture picks up the value of the other from the PowerExchange configuration file.
Oracle Schema		Overrides the schema name for a group of registrations enabling you to use a single set of registrations to capture data from multiple schemas that may exist in a given Oracle instance.
DB2 UDB Database		Overrides the DB2 UDB database connection obtained from the extraction map.
CAPI Connection Name		Overrides the default database connection made in the DBMOVER configuration file. Specify the value given in the NAME parameter of the required CAPI_CONNECTION statement in the DBMOVER configuration file to point to the relevant data source. This value is required only when multiple CAPI connection statements are present in the DBMOVER configuration file.
Extraction Schema		Overrides the extraction schema obtained from the extraction map.
Extraction Map		
Location		If you are extracting change data in continuous extraction mode from PowerExchange Logger for Linux, UNIX, and Windows log files that are remote from the extraction maps on the data source, enter the node name of the system where the extraction maps are located. You must define this node in a NODE statement in the dbmover.cfg file where the PowerExchange Logger for Linux, UNIX, and Windows runs. Usually, this node name is the node name that is specified in the CAPTURE_NODE parameter in the pwxccl.cfg configuration file. For example, if a PowerExchange Logger for Linux, UNIX, and Windows process captures change data from a remote IMS source on an MVS system, specify the node name for the MVS system.

Parameter	Values	Description
UserID		User ID that is used to authorize PowerExchange Logger for Linux, UNIX, and Windows access to extraction maps on the remote node that is specified in the **Location** field. Whether this user ID is required depends on the SECURITY setting in the DBMOVER configuration file at the remote location. This user ID can be either an operating system user ID or database ID, depending on the data source type. Usually, this user ID is the one specified in the CAPTURE_NODE_UID parameter in the pwxccl.cfg configuration file.
Password		A clear text password that is used with the user ID specified in the **UserID** field to authorize PowerExchange Logger for Linux, UNIX, and Windows access to extractions maps at the remote location. Usually, this user ID is the one specified in the CAPTURE_NODE_PWD parameter in the pwxccl.cfg configuration file.

DB2390IMG Advanced Parameters

The DB2390IMG access method creates a data map dynamically as part of its processing. In the DB2390IMG Advanced Parameter dialog box, enter a schema name and data map name for this data map. PowerExchange uses this schema name and data map name to construct the data map name as follows:

```
schemaname.datamapname.dmp
```

In the DB2390IMG Advanced Parameters dialog box, enter the following information:

Parameter	Description
Schema Name	Schema name for the data map. Must begin with an alphabetic character. Maximum length of ten bytes.
Data Map Name	Map name for the data map. Must begin with an alphabetic character. Maximum length of ten bytes.

EMR Advanced Parameters

In the EMR Advanced Parameters dialog box, enter or select the following information:

Parameter	Values	Description
Extract Type	SL,RS	- SL. Extract all data since last extract run. - RS. Restart previous, or specified, extract.
Image Type	BA,AI,TU	- BA. Before and after images are delivered as Delete and Insert records for any change. - AI. Captures only the data after image (the latest change). - TU. Before and After images are delivered with an action indicator of change, rather than Delete and Insert records as delivered by BA. The application that processes the row must be able to apply changes rather than just delete and insert affected rows.
Timeout	0 to 86400	Sets the maximum approximate time in seconds to wait for data on a queue before returning EOF. The following values indicate the following actions: - 0 indicates that EOF is returned immediately that there is no more data. - 86400 indicates that EOF is never returned and the job waits forever.
No Progress Update		Default is selected. If No Progress Update is selected, PowerExchange does not treat this extract as a true application extract.

Parameter	Values	Description
AS400 Library/Journal		Specifies the fully qualified library and journal name to use instead of that specified in the PowerExchange change capture registration. For example: `STQA/NEWJOURNAL`
Oracle Instance		Overrides the Oracle instance information for a given Oracle Collection Id enabling you to use a single set of registrations to capture data from multiple Oracle instances. This overrides the second value of the ORACLEID statement in the PowerExchange configuration file. For example: `ORACLEID=(coll_id,oracle_sid,connect_string, cap_connect_string)` Used in conjunction with the Oracle Connection string. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture determines the value of the other keyword by using the PowerExchange configuration file.
Oracle Connection String		Overrides the Oracle connection information for a given Oracle Collection Id enabling you to use a single set of registrations to capture data from multiple Oracle instances. This value overrides the fourth value of the ORACLEID statement in the PowerExchange configuration file. For example: `ORACLEID=(coll_id,oracle_sid,connect_string, cap_connect_string)` Used in conjunction with the Oracle Instance. The user can specify either or both Instance/Connection string; if one of the keywords is not specified, Oracle Capture picks up the value of the other from the PowerExchange configuration file.
Oracle Collection ID		Enables the PowerExchange Listener to submit multiple simultaneous Oracle capture processes that can connect to different Oracle instances. Oracle Capture uses the override to determine to which instance it should connect, instead of using the ORACOLL keyword of the CAPI_CONNECTION TYPE=ORCL statement in the PowerExchange configuration file. This parameter enables you to use a single PowerExchange Listener to capture data from up to ten Oracle instances simultaneously.
DB2 UDB Database		Overrides the DB2 UDB database connection obtained from the extraction map.

Generating Restart Tokens for Change Data Capture

You can use database row test to generate restart tokens for data sources. You can then use these generated restart tokens to populate the PWXPC restart token file for a PowerCenter session. PowerExchange generates restart tokens using the location and source type you specify. These restart tokens represent the current end of the change stream at the time at which you perform the database row test. The PowerExchange Navigator displays the generated restart tokens in the Database Row Test Output window. You can then copy the output from this window so you can populate the PWXPC restart token file.

The following methods also generate restart tokens:

- The special override statement with CURRENT_RESTART in the PWXPC restart token file. PWXPC and PowerExchange generate restart tokens that represent the current end of the change stream at the time the PowerCenter session runs.
- The DTLUAPPL utility with the GENERATE RSTTKN parameters and a valid capture registration. DTLUAPPL generates restart tokens that represent the current end of the change stream at the time the utility runs.

For more information about these other methods, see *PowerExchange Interfaces for PowerCenter*.

Use the following procedure to generate restart tokens from PowerExchange Navigator database row test.

To generate restart tokens:

1. On the menu bar, click **File > Database Row Test**. The Database Row Test window appears.
2. In the **DB Type** list, select **CAPXRT**.
3. In the **Location** list, select the node name that represents the location of the data source.

   PowerExchange Navigator populates the **Location** list with the node names you specify in the dbmover.cfg file on the Windows machine.
4. In the **Application Name** box, enter a value for the application name.

   You can enter between 1 and 20 characters. PowerExchange does not store this application name so you can enter any value in this box.
5. In the **SQL Statement** box, enter the PowerExchange SQL statement to generate restart tokens. For more information about the syntax of the SQL statement, see "SQL Statement for Generating Restart Tokens" on page 148.
6. Click **Go**.

   The Database Row Test Output window appears and displays the generated restart tokens.
7. To copy the generated restart tokens, right-click in the output window and then click **Copy Output**.

   You can use the generated restart tokens in the PWXPC restart token file. For more information about PWXPC restart token files, see *PowerExchange Interfaces for PowerCenter*.

SQL Statement for Generating Restart Tokens

PowerExchange provides a unique SQL statement to generate restart tokens. Use the following syntax to generate restart tokens:

```
SELECT CURRENT_RESTART [WHERE CONNAME=conn_name|CONTYPE=conn_type]
```

This SQL statement has the following parameters:

SELECT CURRENT_RESTART

Specifies that PowerExchange should generate current restart tokens.

If you do not specify either CONNAME or CONTYPE, PowerExchange uses the CAPI_CONN_NAME value for the PowerExchange Listener in the source location to determine the data source type.

CONNAME=*conn_name*

Specifies the CAPI_CONNECTION name at the source location that PowerExchange uses to determine the data source type.

CONTYPE=*conn_type*

Specifies the connection type that PowerExchange uses to determine the data source type.

Restart token formats vary by source type. Specify one of the following values for *conn_type*:

- **ADA** for Adabas sources (MVS)
- **AS4** for DB2 for i5/OS sources
- **DCM** for Datacom sources (MVS)
- **DB2** for DB2 for z/OS sources (MVS)
- **IDL** for IDMS log-based sources (MVS)
- **IDM** for IDMS synchronous sources (MVS)
- **IML** for IMS log-based sources (MVS)
- **IMS** for IMS synchronous sources (MVS)

- **ORA** for Oracle sources
- **MSS** for Microsoft SQL Server sources
- **UDB** for DB2 for Linux, UNIX, and Windows sources
- **VSAM** or **VSM** for VSAM sources (MVS)

For MVS sources, all MVS connection types result in restart tokens of the same format because PowerExchange records all change data from MVS sources in the PowerExchange Logger. Restart tokens for MVS sources represent locations within the PowerExchange Logger logs.

CHAPTER 9

Application Groups and Capture Applications

This chapter includes the following topics:

Application Groups and Capture Application Overview

This function provides the capability for the user to display the information for a given extract application and to allow the user to reset the status of the extractions process to rerun.

This shows an audit trail of all the extractions that have been run, their status and an indication of the captured data that has been moved.

To view the application restart information open the Application Group by right clicking and selecting **Open** or double clicking the Application Group.

All applications appear in the list in the Application Group and not just those associated with the Registration Group created at the same time. When the applications are open, it shows a history of the change extracts.

The Application Group is based on the same principle as the Extraction and Registration Groups. The Group level identifies the location of the PowerExchange Listener task. There is no need for a database instance type or name. The Resource Inspector shows:

- Name
- Location
- Userid
- Password

The Application Name list is based on the application name on the extraction run.

Clicking on a specific Application Name in the list displays a list of all the extracts (successful or not) in timestamp order, latest one on top. Click the timestamp to display the complete information for that extraction run, including registration name, table name, and timestamps.

Right-clicking on a specific Application Name displays a list of possible actions for that application name.

Note: Informatica recommends that a unique application name be used for each extraction process. Using unique application names for each extraction process ensures that proper restart occurs. Sharing application

names with multiple extraction processes can result in errors, extraction of undesired change data, and other unexpected results.

Resetting the Start Point for an Application

The Application tracks the extractions against a registration and stores the start point for the next run. Occasionally, one or more extractions must be rerun and you can achieved this by resetting the start point for the application.

This feature is used to reset the start point of the next extraction run for the selected application. It resets the start point to the start point of the open timestamp.

The application can then either:

- Restart at the start of the selected run, or
- Be reset again.

This feature is powerful and cannot be undone. If in doubt, use DTLUAPPL. Tokens can be printed for copy or paste using DTLUCDEP.

The following example uses an application DB2TEST, which has had two extractions run against it. This example, for demonstration purposes only, shows extract times that are close together. However, the application shows how to reset the start point.

To reset the start point:

1. Open the required application by double clicking on the application group then the relevant application.

   Note: The order of runs shown in the right hand pane. The first and last runs show the time of the two runs and the Current Run is blank. Highlight the run to which the start point is to be reset, in this case the earliest of the two runs is selected.

2. Select the Application Group name, right clicking and selecting **Reset To New Start Point**.

   The icon to the left of the application group turns red. When exiting there a prompt to save appears.

   The restart point is reset to the start point of the first run. When next opening the application it can be seen that the Current Run information is now no longer blank but has been replaced by the run information from the first extraction.

CHAPTER 10

Working with User-Defined Fields

This chapter includes the following topics:

User-Defined Fields Overview

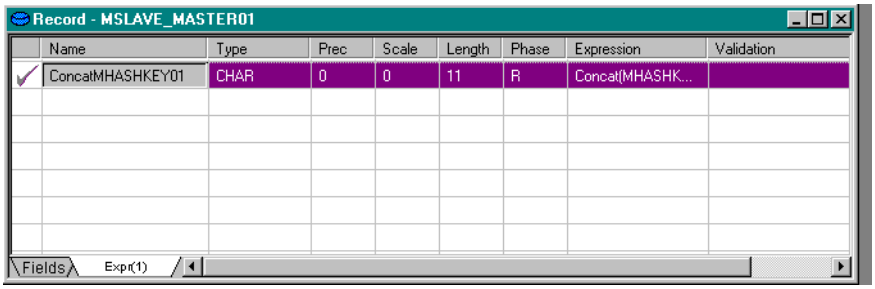
The purpose of user-defined fields is to add the flexibility of manipulating data from the source and creating value added data maps. You can create fields by using one or more functions defined as expressions:

- Calls to customer written COBOL, PL/1, Assembler, or C programs.
- Calls to PowerExchange internal functions, such as concatenation and field splitting.

Expressions Tab

The Expr(x) tab of the Record window contains the user-defined fields that have been set-up for the current data map. Each row contains a declaration of a field or an assignment of a particular value to a previously declared field. The number x indicates the number of expressions that have been defined. For example, Expr(0) indicates that none have been defined while Expr(6) indicates that 6 have been defined.

The following figure shows a user-defined field for a Data Map:



The following table describes the columns in the user-defined fields window:

Column	Description
Field status indicator	Field entered correctly and is operable or a problem exists with the field. Read the comment in the Validation column. New field selected (after selecting one of the Add Field ... menu options).
Name	Name given to the user-defined field, which identifies the new field.
Type	Datatype of the new field.
Prec	Precision of the new field.
Scale	Scale of the new field.
Length	Length of the new field.
Phase	R (Read), W (Write), or RW (Read and Write).
Expression	You can enter the expression that is used to populate the field.
Validation	If there is a problem with the field, a message appears.

Right-Click Menu

The following menu is available by right-clicking the mouse.

Column	Description
Add Field Before ...	Adds a new field row above the current item.
Add Field After ...	Adds a new field row below the current item.
Add Field at End ...	Adds a new field row at the bottom of the list.
Move Field Up	Moves declarations to the top of the list and assignments to the bottom.
Organise Fields	Moves all completed new fields to the top of the list.
Delete Field/Cell	Deletes the current cell or field, if the entire row is selected.
Hide/Show Type Columns	Hides or shows the Type, Pre(cision), Scale, and Length columns.
Restore Columns	Shows all columns and restores all columns to the default width.

Declaration

The row contains the new field name, a datatype with appropriate precision, scale and length values and the phase of database access in which the field operates: Read, Write, or Read/Write.

Assignment

The assignment row must contain the name of the previously declared field, the phase of database access in which the field operates and an expression which assigns a particular value to the field. It is possible to combine the declaration and assignment for a new field by entering an expression on the declaration line.

Expression

The expression determines what action is performed on the field to populate it with data. The expression can be:

- A text or numeric constant, such as 'My text' or 1234.
- A predefined PowerExchange function, such as concatenation or field splitting.
- A call to a user-defined external program, such as COBOL, PL/1, Assembler, or C programs.

Phase

The phase is the aspect of database communication for which the user-defined field is appropriate. For example, if a field is defined as having a phase of Read, its expression is not recalculated when the database is being written to.

Keyboard Shortcuts

The following keyboard shortcuts are available.

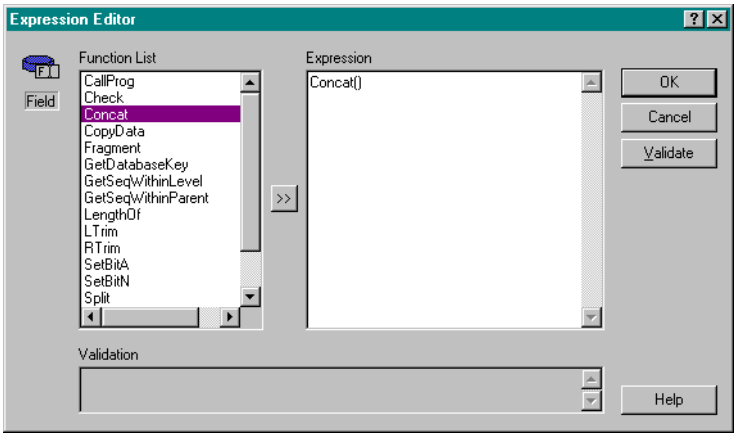
Key	Description
Insert	Insert a new row at the bottom
Delete	Delete field or cell contents depending on whether a cell or row is highlighted
Enter	Start editing a cell or enter contents from editing cell
Escape	Leave a cell without making changes
Ctrl+Up Arrow Key	Moves a field up
Ctrl+Down Arrow Key	Moves a field down
Arrow keys	Moves highlighted cell around

Expression Editor Dialog Box

This dialog box allows you to build the expression that is associated with the field. Double-click the appropriate function in the Functions List and then manually edit the parameters to build the expression.

The dialog box is accessed by clicking the Browse button in the Expression column of the Expr(x)s tab on the Record Properties window.

The following figure shows the Expression Editor dialog box:



The following table describes the fields in the Expression Editor dialog box:

Object	Description
Expression	You can enter the expression that is used to populate the field.
Validation	If there is a problem with the field a message appears after clicking Validate.
Function List	Lists the available functions from which you can build and expression the Expression box. Double-click a function to add it to the current expression.
OK	Closes the dialog box and associates the expression with the new field.
Cancel	Cancels any actions that you have performed and closes the dialog box.
Validate	Validates the expression that you entered in the Expression box. If the validation produces an error, the error message appears in the Validation box.

Creating a User-Defined Field

The following procedure enables you to create a new field and associate a custom definition to it.

To create a user-defined field:

1. In the Resource Explorer panel select the required data map.
2. Select the record. The Record window appears.
3. Click the Expr(x) tab of the Record window.
4. Right-click and select Add Field at End.
5. Enter a name for the new field in the Name cell.
6. Select the datatype from the Type cell.
7. Enter the Pre(cison), Scale, and Length as appropriate to the datatype selected.
8. Enter the expression that you wish to associate with the field. You can do this by directly typing into the cell or by clicking the Browse button.

   The Expression Editor dialog box is displayed.
9. Double-click the appropriate function and edit the parameters manually in the Expression box.
10. Click **Validate**.

    Any errors in the expression are displayed.
11. Click **OK**.

    The expression is associated with the field and the dialog box is closed.
12. To view the user-defined field, right-click the table and select **Properties**. In the Table Properties dialog box, select **Refresh with missing columns** in the **Column Generation** list.

Available Functions

The following predefined functions are included.

CallProg

Purpose

Calls a user-defined sub-routine or program.

Syntax

```
CallProg('Module','Routine', 'Linkage', arg1 ..."
```

Parameters

Name	Description
Module	Module where the subroutine resides, which is the physical name of the program. - On i5/OS, a service program. - On MVS, a load module. - On Windows, a DLL. - On UNIX, a shared object.
Routine	Name of entry point. - On i5/OS, Windows, and UNIX, this name is the subroutine name. - On MVS linkages for types of Assembler, COBOL, a value must be provided but it is ignored and the default entry point for the load module is used. It is best to repeat the module name. - For MVS PLI, it is sometimes possible to have multiple fetchable subroutines in the same load module.
Linkage	Argument determining the type of linkage which has two factors: - The method by which arguments are passed (C uses the stack, other languages pass a list of addresses. - The parameters are passed (C and OS gets the failure code from the subroutine return code; other types pass the address of a Failure code integer). Supported values are: - C on all platforms. - COBOL on MVS. - PLI on MVS. - OS on MVS. - OS400 on i5/OS.
arg1	First argument passed to external subroutine.
arg2	Second argument passed to external subroutine.

Result

The optional Result is a NUM32. 0 indicates success. Non-zero indicates failure.

If no result argument is specified and a non-zero failure code is returned from the external function, then CALLPROG applies default error handling. By default, any non-zero failure code returned from the external sub-routine triggers a map-level response (either; terminate the extract or skip this routine).

Notes

For more information about the programming interface and return code checking for CALLPROG programs, see “Using External Programs with CALLPROG” on page 171.

Check

Purpose

Provides a limited override facility for reporting errors signalled by external programs that have been called using the CALLPROG function.

By default, any non-zero failure code returned from the external sub-routine triggers a map-level response (either; terminate the extract or skip this routine).

The CHECK function can be used to allow specified non-zero return codes to be tolerated.

Syntax

```
Check(Field1, Message, Comparison, Value1, Value2, …)
```

Parameters

Name	Description
Field1	Field to be checked.It receives the result from CALLPROG.
Message	A user-defined message on the log, such as 'Unrecognized name in program XYZ.'
Comparison	The following types of comparison can be used: - EQ. A failure is signalled if the value in Field1 matches any of values in the list. - NE. A failure is signalled if the value in Field1 does not match any of values in the listString value must be in quotes.
Value1	The first value on the list.
Value2	The second value on the list.
Any number of further parameters can be added.	

Result

No result variable is defined. If the function determines either:

- that the processing for this row should abort or
- continue by setting a system return code.

Example of accepting non-zero return codes

If failure codes of 0, 3, and 4 are required to be acceptable then define:

```
CallProgRC = CallProg('Program1', 'Program2', 'COBOL', 'Field1', 'Field2')
Check(CallProgRC, 'Error in Program1', NE, 0, 3, 4 )
```

Concat

Purpose

Concatenates two or more fields (real or user-defined). This function could, for example, be used to create a group field from several other fields.

Syntax

```
Concat(field1,field2,…)
```

Parameters

Name	Description
Field1	1st field to be copied. Can be a real or user-defined expression field
Field2	2nd field to be copied. Can be a real or user-defined expression field

Example

If the fields (field1) contained the text The quick brown fox and the field (field2) contained the text jumped over the lazy dog, the function Concat(field1,field2) would return the text The quick brown fox jumped over the lazy dog.

Result

The result field is a group field created by copying the input fields together. It must be a User-defined expression field.

Notes

No conversions are done. The number of bytes to copy is determined from the data length.

Copydata

Purpose

Copies data from the specified field into the new user-defined field.

Syntax

```
CopyData(SourceFieldName)
```

or

```
CopyData(SourceFieldName,TargetFieldName)
```

Parameters

Name	Description
SourceFieldName	Input field to be copied. Can be a real or user-defined expression field
TargetFieldName	Specifies the target field to copy to (if Result is not specified). The target field can only by a user-defined expression field.

Result

The result field is optional. If used, it specifies the target field for the move and the 2nd argument is not given.

Notes

- The target field can either be specified in the Result field using prototype the first prototype or it is specified as the second argument as in the second form of syntax shown previously.
- When an expression evaluates to a field, a call is made to this function.

Fragment

Purpose

Splits the contents of an existing field and distributes the result among two or more other existing fields.

Syntax

```
Fragment(SourceField,TargetField1,TargetField2, …)
```

Parameters

Name	Description
SourceField	Field whose data is to be fragmented
TargetField1	The first target field
TargetField1	The second target field

Example

Pre-existing fields:

```
Field1 (44)
TargetField1 (15)
TargetField1 (29)
```

If the field (Field1) contained the text The quick brown fox jumped over the lazy dog, then the function Fragment(Field1, TargetField1, TargetField2) would return:

```
TargetField1 = "The quick brown"
TargetField2 = " fox jumped over the lazy dog"
```

GenVRowKey

Purpose

This function is used with source rows that contain an array. For data rows with an OCCURS clause, a single row is returned for each instance of the OCCURS.

The function returns a generated unique row number for the row which was created from a source row that contained the array.

Syntax

```
GenVRowKey()
```

Result

NUM32U

Example

Assume you have a data map with an array defined by a COBOL OCCURS clause as follows:

```
01 RECORD.
   03 FIELD1 PIC X(4).
   03 FIELD2 PIC X(4) OCCURS 3 TIMES.
```

For 2 source rows containing the following data:

```
AAAA111122221111
BBBB555566667777
```

the following six table rows are generated:

```
AAAA 1111 1
AAAA 2222 2
AAAA 1111 3
BBBB 5555 1
BBBB 6666 2
BBBB 7777 3
```

Note that this allows you to create unique instances of the two rows containing the value 'AAAA 1111'.

GetDatabaseKey

Purpose

Returns the internal database key for the associated record instance.

Syntax

```
GetDatabaseKey()
```

Result

Applicable Data Source	Type	Key object returned
ADABAS	NUM32	Internal Sequence Number (ISN)
ESDS	BIN 8	Extended Relative Byte Address (XRBA)
i5/OS	NUM32	Relative Record Number (RRN)
IDMS	NUM32	Database key
IMS	BIN 8	Relative Byte Address (RBA)
ODBA	BIN 8	Relative Byte Address (RBA)
RRDS	NUM32	Relative Record Number (RRN)

Notes

GetDbKey can be used as an alias for this function when being used with IDMS.

IMS Log-based Change Capture is not currently supported.

GetDataFlowType

Purpose

This describes the type of record returned. This function can be used with both PowerExchange Bulk and Capture processing.

When used with `GetDbKeyOfOwner`, a relational view of the IDMS data can be generated. Records display the owner record key. This data could be loaded into a relational database using PowerExchange Bulk processing, and then kept current by using PowerExchange capture.

Syntax

```
GetDataFlowType()
```

Result

The type of data included in the returned row. The result argument must be a CHAR 2.

The first character can be:

- `B`. Bulk data extraction.
- `L`. Log capture data extraction.
- `R`. Real-time capture data extraction.

The second character can be:

- `D`. Normal data record.
- `C`. Connect without data (relevant for capture only).
- `U`. Disconnect without data (relevant for capture only).

When the second character returned is a `C` or `U`, the data portion of the record returned is filled with null values.

If there is no owner, the value that was returned by `GetDbKeyOfOwner` is set to high values.

Notes

This function is only relevant for IDMS data access.

This function only produces results for a single record capture registration or for the base record (lowest level) of a multiple record registration.

For owner key functionality only the database key is available, the page group and radix elements are always set to zero.

GetDbKey

Alias to GetDatabaseKey

GetDbKeyOfOwner

Purpose

This function returns the database key of the specified owner record. This function must be used only where there is no possibility of duplicate database keys because this function does not return either the IDMS page group or radix. If this is not the case you may need to use GetFullDbKeyOfOwner.

Use of the GetDbKeyofOwner function enables the specified owner record to be identified without physically reading that record.

Syntax

```
GetDbKeyOfOwner('<SETNAME>')
```

Result

The database key. The result argument must be a NUM32U or BIN4.

Notes

This function is only relevant for IDMS data access.

This function cannot be used with a setname which represents a system index.

GetFullDbKey

Purpose

This returns the fully qualified database key of the IDMS record. The use of the fully qualified option may be a requirement when duplicate database keys are present in the database. The unique record is identified by the addition of the four byte concatenated page group and radix identifier prefix.

This adds a four byte overhead to the records passed.

Syntax

```
GetFullDbKey()
```

Result

The database key. The result argument must be a BIN8.

Notes

This function is only relevant for IDMS data access.

GetFullDbKeyOfOwner

Purpose

This returns the fully qualified database key of the owner record. The use of the fully qualified option may be a requirement when duplicate database keys are present in the database. The unique record is identified by the addition of the concatenated four byte page group and radix identifier prefix.

This adds a four byte overhead to the records passed.

Syntax

```
GetFullDbKeyOfOwner('<SETNAME>')
```

Result

The database key. The result argument must be a BIN8.

Notes

This function is only relevant for IDMS data access.

This function cannot be used with a setname which represents a system index.

GetIMSRBAByLevel

Purpose

Returns the RBA value of the requested segment.

Syntax

```
GetIMSRBAByLevel(integer)
```

Parameters

Name	Description
integer	Specify one of the following values: - No value, such as `GetIMSRBAByLevel()` If you do not specify a value, the result is the RBA of the current segment - Positive integer, such as `GetIMSRBAByLevel(2)` Input is the 4-byte integer level number of the segment in the IMS hierarchy whose RBA you wish to access. The level number is limited to the level number of the current segment in the hierarchy, or any of that segments ancestors. In other words, if you are dealing with a segment at level 3 of a 5-level hierarchy, you may request the RBA of segments at level 1, 2, or 3. Attempting to request the RBA of segments in the hierarchy that are below the level of the current segment results in a run time error. - Negative integer, such as `GetIMSRBAByLevel(-2)` You may also make a relative request by providing a negative number as input. For instance, you may request the RBA of the segment two levels above the level of the current segment. You do this by specifying -2. An attempt to specify a negative number whose absolute value is greater than or equal to the level number of the current segment results in a run time error.

Result

Output of GetIMSRBAByLevel is an 8-byte binary representation of the requested segment's RBA value. 8 bytes are used to allow for future expansion and for a uniqueness guarantee for FastPath and HALDB databases.

Notes

GetIMSRBAByLevel is a standard PWX field, and can be used for Bulk and synchronous Change Capture IMS maps.

IMS Unload files and IMS Log-based Change Capture are not currently supported.

GetPageGroup

Purpose

Returns the Page Group within which the IDMS record exists.

Syntax

```
GetPageGroup()
```

Result

The Page Group. The result argument must be a Num16, Num16U, or Bin(2).

Notes

This function is only relevant for IDMS data access.

GetPgGrpAndRdx

Purpose

Returns the Page Group and Radix of the IDMS record.

Syntax

```
GetPgGrpAndRdx()
```

Result

The concatenated Page Group and Radix. The result argument must be a Num32, Num32U, or Bin(4).

Notes

This function is only relevant for IDMS data access.

GetPgGrpOfOwner

Purpose

Returns the Page Group of the Owner Record.

Syntax

```
GetPgGrpOfOwner('<SETNAME>')
```

Result

The Page Group of the owner record. The result argument must be a Num16, Num16U, or Bin(2).

Notes

This function is only relevant for IDMS data access.

GetRadix

Purpose

Returns the Radix of the IDMS record.

Syntax

```
GetRadix()
```

Result

The Radix. The result argument must be a Num16, Num16U, or Bin(2).

Notes

This function is only relevant for IDMS data access.

GetRadixOfOwner

Purpose

Returns the Radix of the Owner Record.

Syntax

```
GetRadixOfOwner('<SETNAME>')
```

Result

The Radix of the owner record. The result argument must be a Num16, Num16U, or Bin(2).

Notes

This function is only relevant for IDMS data access.

GetPgGrpAndRdxOfOwner

Purpose

Returns the Page Group and Radix of the Owner record.

Syntax

```
GetPgGrpAndRdxOfOwner('<SETNAME>')
```

Result

The concatenated Page Group and Radix of the owner record. The result argument must be a Num32, Num32U, or Bin(4).

Notes

This function is only relevant for IDMS data access.

GetSeqWithinLevel

Purpose

Returns the record sequence number within the current hierarchical level.

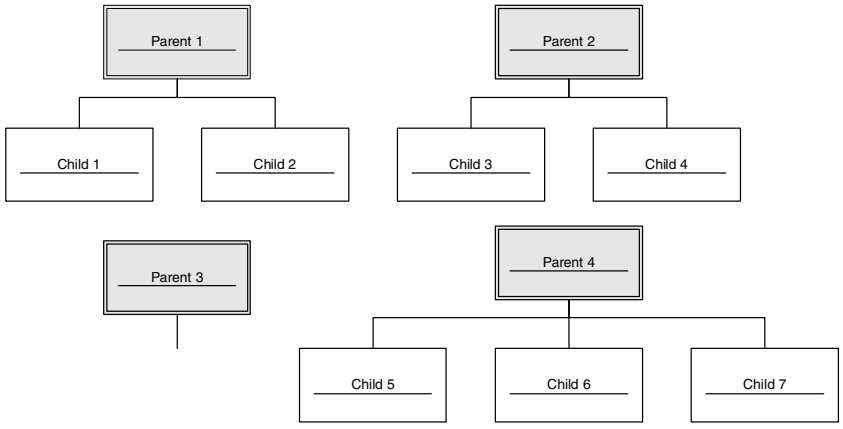
Syntax

```
GetSeqWithinLevel()
```

Result

The record sequence number. The result argument must be a NUM32.

Example



The function GetSeqWithinLevel returns the record sequence numbers:

```
1,2,3,4,0,5,6,7
```

The record sequence numbers reflect the following records:

```
Children 1 and 2 from Parent 1
Children 3 and 4 from Parent 2
Zero (no children) from Parent 3
Children 5, 6 and 7 from Parent 4
```

GetSeqWithinParent

Purpose

Returns the record sequence number relative to the current parent record instance.

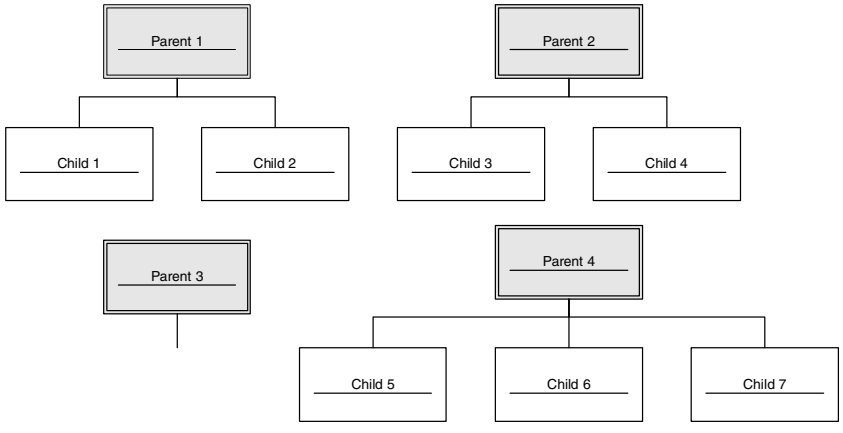
Syntax

```
GetSeqWithinParent()
```

Result

The record sequence number. The result argument must be a NUM32.

Example



The function GetSeqWithinParent returns the following record sequence numbers:

```
1,2,1,2,0,1,2,3
```

The record sequence numbers reflect the following records:

```
Children 1 and 2 from Parent 1
Children 1 and 2 from Parent 2
Zero (no children) from Parent 3
Children 1, 2 and 3 from Parent 4
```

LengthOf

Purpose

Returns the length of the given field.

Syntax

```
LengthOf(fieldname)
```

Parameters

Name	Description
fieldname	The NAME Of the field whose length is being retrieved

Result

The length of the given field. The result argument must be a NUM32.

Notes

This function is only relevant for field types where the data length varies, such as STRING, VARCHAR, or CHAR.

Example

If the field (field1) contained the text brown fox, the function LengthOf(field1) would return 9.

LTrim

Purpose

Returns a CHAR value which contains the original field value with all character values stripped from the left hand end.

Syntax

```
LTrim (field, character)
```

Parameters

Name	Description
Field	The field to be trimmed.
Character	The character parameter is a single character which must be enclosed in single quotes. If no character parameter is entered, then a 'space' is the default.

Result

CHAR

Example

If the field Text contains the text “$$$$$$$$$The quick brown fox,” the function LTrim(Text,'$') returns the text “The quick brown fox.”

RTrim

Purpose

Returns a CHAR value which contains the original field value with all character values stripped from the right hand end.

Syntax

```
RTrim (field, character)
```

Parameters

Name	Description
Field	The field to be trimmed.
Character	The character parameter is a single character which must be enclosed in single quotes. If no character parameter is entered, then a 'space' is the default.

Result

CHAR

Example

If the field Text contains the text “The quick brown fox**********,” the function LTrim(Text,'*') returns the text “The quick brown fox.”

Split

Purpose

Returns the portion of the given field that is delimited by the start point and number of characters from the start point.

Syntax

```
Split(Field,Start,Number_Bytes)
```

Parameters

Name	Description
Field	Field whose data length is being retrieved
Start	First byte to copy, where numbering starts at 1
Number_Bytes	Number of bytes to copy

Result

Result argument gets a subset of bytes from Field, starting at the Start byte.

Notes

This function can be used, for example, to get a single field from a group field. The numeric arguments must be NUM32 field types.

The Split function does not cater for datatype changes. For example, it may be necessary to split off part of a source character field to create a numeric target field. You should use the Split function on the source field to create a new character field; then use the Copydata function on the newly-created character field to create the target field with a numeric datatype.

Example

If the field (field1) contained the text “The quick brown fox jumped over the lazy dog”, then the function Split (field1, 17, 10) would return the text “fox jumped”.

Strip

Purpose

Returns a CHAR value which contains the original field value with all character values stripped from either the left hand end, right hand end or both ends depending on the side parameter.

Syntax

```
Strip (field, side, character)
```

Parameters

Name	Description
Field	The field to be trimmed.
Side	The side parameter is L,R, B. or space which must be enclosed in single quotes. If space is selected, then B is the default.
Character	The character parameter is a single character which must be enclosed in single quotes. If no character parameter is entered, then a 'space' is the default.

Result

CHAR

Example

If the field Text contains the text “**********The quick brown fox**********,” then the following functions return the following text:

- Strip(Text, 'R', '*') returns the text “**********The quick brown fox.”
- Strip(Text, 'L', '*') returns the text “The quick brown fox**********”
- Strip(Text, 'B', '*') returns the text “The quick brown fox.”

SetBitA

Purpose

Returns a CHAR value of Y or N depending on the bit setting of field at the offset position.

If the bit is set 'on' then Y is returned, if the bit is set 'off' then N is returned.

Syntax

```
SetBitA (field, offset)
```

Parameters

Name	Description
Field	The field to be checked.
Offset	The position within the field of the bit that is being checked. The offset begins at bit 0 and ends at bit 7 for a 1 byte field, 15 for 2 bytes, and so on.

Result

CHAR

SetBitN

Purpose

Returns a numeric (NUM8) value of 1 (on) or 0 (off) on the bit setting of field at the offset position.

Optionally, you can override the result by setting the optional parameters on_value and off_value. For example you could set an on_value to 3 and an off_value to 2. If the bit being tested is set 'on' then on_value (3) is returned rather than 1; if the bit is set 'off' then off_value (2) is returned rather than 0.

Syntax

```
SetBitN (field, offset, on_value, off_value)
```

Parameters

Name	Description
Field	The field to be checked.
Offset	The position within the field of the bit that is being checked (1 to 8).
on_value	Set to override the default bit value of 1. Returned if the bit being checked is set 'on'. Optional.
off_value	Set to override the default bit value of 0. Returned if the bit being checked is set 'off'. Optional.

Result

NUM8

ToLower

Purpose

Returns a lower case version of the contents of the field.

Syntax

```
ToLower (fieldname)
```

Parameters

Name	Description
Fieldname	The field to be converted. Valid only for CHAR and VARCHAR fields.

Result

CHAR and VARCHAR

ToUpper

Purpose

Returns an upper case version of the contents of the field.

Syntax

```
ToUpper (fieldname)
```

Parameters

Name	Description
Fieldname	The field to be converted. Valid only for CHAR and VARCHAR fields.

Result

CHAR and VARCHAR

Using External Programs with CALLPROG

You can use the CALLPROG function to call user-written subroutines or programs. For example, you can use CALLPROG to invoke user-written programs to decode or supplement data in specific fields.

Use the following syntax for the CALLPROG expression:

```
CallProg('Module','Routine', 'Linkage', 'arg1', 'arg2', ...)
```

Invoking External Programs

The function address for the entry point of the external routine is loaded on the first call. The load may fail for the following reasons:

- The program is not found in the standard order of search for programs on the platform.
- The program is not of the required type:
  - i5/OS. Service program.
  - MVS. Load module.
  - Windows. DLL.
  - Linux and UNIX. A shared object.
- The module handle is obtained but the routine is not found within it. This can indicate platform-specific linking or symbol export problems such as:
  - i5/OS. Symbol for routine not exported.
  - MVS C or PL/I. Routine not defined with #pragma fetchable.
  - Windows C. Routine not exported.
  - On some platforms, routines are case-sensitive.

Parameter List Passed to External Programs

PowerExchange passes the following parameters to programs invoked by the CALLPROG function:

Name	Description

NbrFlds	Number of fields.
	An integer specifying the number occurrences of fields in the arrays ppData[] and ppData[].
	Input usage only.
	If 2 fields are being passed and the expression entered by the Navigator is
	`CallProg('Module','Routine','Linkage',Field1,Field2)`
	then NbrFlds is set to 2.
	The usage of fields
	`int NbrFlds, char *ppData[],int * pDataLen[]`
	is similar to the C usage of
	`main(int argc, char *argv[]).`

pMsgBuffer	Message Buffer. A pointer to character string where the external routine can pass back a NULL terminated string to describe an error, which accompanies the setting of the return code to a non-zero value. Output usage only. The message buffer is always empty on every call to the external function. If no error is met, the external function should leave it untouched.

pMsgBufferSize	Message Buffer Size.
	Usage is input and output.
	Pointer to an integer giving the size of the Message Buffer. If an error is met, the external routine can return an error message describing the error.
	For example using syntax
	`strncpy(pMsgBuffer,"Error in program FRED",`
	`(*pMsgBufferSize) -1);`
	
	Ensure that you do not exceed the memory buffer allocated by the CALLPROG routine. Currently this buffer is 128 bytes, but this may change in the future, so the pMsgBufferSize should be used.
	
	The actual size of the message returned to the caller can be optionally returned to CALLPROG. It is only needed if the returned string is not NULL terminated. The returned message is only used when the return code is not zero.

ppData[]	Array of pointers to data characters.
	Usage is input and output.
	The data for a field is accessed through this pointer in a similar way to the way that the argv is used on the C main function.
	Example
	To get the data from the 2nd field of field type STRING:
	`char Name[50];`
	`...`
	`strcpy(Name, ppData[1]);`

pDataLen[]	Array of pointers to integers.
	Usage is input and output.
	It determines the length of the field.
	Its importance depends on the field type.
	For fixed-length field types, such as DOUBLE, FLOAT, NUM8, NUM16, NUM32, NUM64 DATE, and TIME TIMESTAMP, the field always has the same size if it is present, or has a zero size if it was NULL.
	For variable length types which are not delimited by nulls, such as VARCHAR, the actual length of the data can only be determined from pDataLen.
	Where fields are returned to CALLPROG, the data length must be reset if it has changed.
	For example, to change the data and length of 2nd field of field type VARCHAR:
	`char Name[50];`
	`...`
	`strcpy(ppData[1], Name);`
	`*pDataLen[1] = strlen(Name);`

Result Argument and Error Handling

By default, the result argument from CALLPROG does not need to be defined on the Navigator screen. In this case default error handling applies, that is, any non-zero failure code returned from the external sub-routine triggers a map-level response. The map-level response is either terminate the extract or skip this routine.

The function CHECK provides a limited facility to override the errors that are reported. For example, if failure codes of 0, 3, and 4 are acceptable, then define:

```
CallProgRC = CallProg('Program1', 'Program2', 'COBOL', 'Field1', 'Field2')
Check(CallProgRC, 'Error in Program1', NE, 0, 3, 4 )
```

Return codes

- If the external routine completes successfully, it should return 0.
- In the event of a failure, it should return a non-zero value that can be positive or negative.

  By default, any non-zero failure code returned from the external sub-routine triggers a map-level response (either; terminate the extract or skip this routine).

External C Routines

PowerExchange provides sample C code for a CALLPROG program in the UCPE member of the *hlq*.SRCLIB library, where *hlq* is the high-level qualifier that you specified during PowerExchange installation.

Platform Support

Use linkage type C in the third argument to CALLPROG expression to link to a C language program.

PowerExchange supports external C routines on all platforms and has the following requirements:

- On i5/OS, routines must be exported routines in service programs.
- On MVS, routines must be fetchable from a load module. Called programs must be linked with AMODE(31). If access is required to modules linked AMODE(24), the routine must be called through an AMODE(31) program that relocates the parameters below the 16 MB line and then calls the AMODE(24) load module.
- On Windows, routines must be exported routines in the CDECL convention in a DLL.
- On Linux and UNIX, routines must be exported routines in an executable shared object.

C Function Prototype

```
int CRoutine(int NbrFlds
      ,char *pMsgBuffer, int *pMsgBufferSize
      ,char *ppData[], int * pDataLen[]);
```

For more information about the parameters passed to programs invoked from CALLPROG, see "Parameter List Passed to External Programs" on page 172.

External COBOL Routines

PowerExchange provides sample COBOL code for a CALLPROG program in the UCPEC and UCPEC01 members of the SRCLIB library.

Platform Support

Use linkage type COBOL in the third argument to CALLPROG expression to link to a COBOL language program. COBOL programs can be used on MVS only.

COBOL programs should be compiled using the Language Environment (LE) COBOL run-time routines. Results are unpredictable for COBOL programs compiled with non-LE COBOL run-time routines.

COBOL Linkage

In COBOL, the first four arguments are always the same and are mandatory.

The remaining fields vary according to the needs of the external program. The attributes of the user-defined expression fields must match the parameters that the external program expects.

Example of COBOL Linkage

```
003700 LINKAGE SECTION.
003800
003900 01  NUMBER-FIELDS           PIC S9(9) COMP.
003901
003902 01  FAILURE-CODE            PIC S9(9) COMP.
003903
003904 01  MESSAGE-BUFFER.
003905     05  MESSAGE-BUFFER-BYTE PIC X(1)
003906                 OCCURS 1 TO 128
003907                 DEPENDING ON MESSAGE-BUFFER-LENGTH.
003908 01  MESSAGE-BUFFER-LENGTH   PIC S9(9) COMP.
003910
004000 01  TEXT-AREA.
004010     05  TEXT-AREA-BYTE      PIC X(1) OCCURS 15.
004100 01  TEXT-AREA-LENGTH        PIC S9(9) COMP.
004111
004120 01  NUMBER1                 PIC S9(9) COMP.
004130 01  NUMBER1-LENGTH          PIC S9(9) COMP.
005730
005800 PROCEDURE DIVISION USING
005801     NUMBER-FIELDS
005802     FAILURE-CODE
005803     MESSAGE-BUFFER
005807     MESSAGE-BUFFER-LENGTH
005809     TEXT-AREA
005810     TEXT-AREA-LENGTH
005811     NUMBER1
005820     NUMBER1-LENGTH
```

In the sample program UPEC, two fields are passed to the COBOL program.

- The first field is called TEXT and has a maximum size of 15 bytes. Data can be moved from and to it using COBOL field TEXT-AREA. If it is not a fixed-length CHAR field, then the TEXT-AREA-LENGTH must be used to determine the actual length. If the COBOL program wants the length to change, it must store the required value in field TEXT-AREA-LENGTH.
- The second field is numeric, which in the Navigator is defined as NUM32. Data can be moved from and to it using COBOL field NUMBER1. It would be unusual to want to make use of field NUMBER1-LENGTH.

  If the field is not nullable, then the field would contain value 4 on entry to the COBOL program which should leave the value untouched. NUMBER1-LENGTH is set to zero if the field is NULL. This might happen if the field was NULL before the program was called. If the program decides to make the field NULL, it move zero to NUMBER1-LENGTH.

External OS Routines

PowerExchange provides sample Assembler code for a CALLPROG program in the UCPEA member of the SRCLIB library.

Platform Support

Use linkage type OS in the third argument to CALLPROG expression to link to an Assembler language program. Assembler language programs can be used on MVS only.

Assembler programs receive a list of parameter addresses in a parameter list pointed to by Register 1. An Assembler program should place the return code in register 15.

Assembler Linkage

In Assembler, the first three arguments are always the same and are mandatory. Usage of the arguments and return code is similar to that used for C and COBOL routines.

The definitions for the fields vary according to the number of fields and their types.

Example Linkage

```
L     R3,0(R1)                get address of argument 1
         L     R4,0(R3)                get NumberFields value
         C     R4,=F'2'                Required value of 2 ?
         BNE   BADARGS
*-------------------------------------------------------------------
* Get arguments
*-------------------------------------------------------------------
NBRARGOK DS     0H
         L     R3,4(R1)                get address of argument 2
         ST    R3,AMSGBFF              = address of message buffer
         L     R3,8(R1)                get address of argument 3
         ST    R3,AMSGBFSZ             = size of message buffer
         L     R3,12(R1)               get address of argument 4
         ST    R3,ATEXT                = address of text argument
         L     R3,16(R1)               get address of argument 5
         ST    R3,ATEXTLEN             = length  of text argument (15)
         L     R3,20(R1)               get address of argument 6
         ST    R3,ANUMBER              = address of number
         L     R3,24(R1)               get address of argument 7
         ST    R3,ANUMBLEN             = length  of number (always 4)
...
AMSGBFF  DS    F
AMSGBFSZ DS    F
ATEXT    DS    F
ATEXTLEN DS    F
ANUMBER  DS    F
ANUMBLEN DS    F
```

In the example program, two fields are passed to the Assembler program. The first field is called TEXT. Data can be moved to it and from it using the fourth address in the list - the address in 12(R1). If it is not a fixed-length CHAR field, then use the data length to determine the actual length the address in 16(R1).

External OS400 Routines

PowerExchange provides sample RPG code for a CALLPROG program in the UCPERPGLE member of the *dtllib*/RPGLE, where *dtllib* is the PowerExchange software library that you specified during PowerExchange installation.

Platform Support

PowerExchange supports C and CL programs on the i5/OS.

Use linkage type OS400 in the third argument to CALLPROG expression to link to an non-C language programs. The OS400 linkage type is required for any language other than C, i5/OS, such as CL, COBOL, PL/I, and RPG. Linkage type OS400 can be used on i5/OS only.

CL Linkage

The first four arguments are always the same and are mandatory. Usage of the arguments and return code is similar to that used in the other languages.

The definitions for the fields vary according to the number of fields and their types.

Example CL Linkage

```
PGM PARM(&NBRFLDS &RC &MSGBUFF &MSGBUFFSZ &TEXT &TEXTLEN &NUM &NUMLEN)
/*-------------------------------------------------------------------*/
/* ARGUMENTS                                                         */
/*-------------------------------------------------------------------*/
             DCL        VAR(&NBRFLDS) TYPE(*CHAR) LEN(4)
             DCL        VAR(&RC) TYPE(*CHAR) LEN(4)
             DCL        VAR(&MSGBUFF) TYPE(*CHAR) LEN(128)
             DCL        VAR(&MSGBUFFSZ) TYPE(*CHAR) LEN(4)
             DCL        VAR(&TEXT) TYPE(*CHAR) LEN(15)
             DCL        VAR(&TEXTLEN) TYPE(*CHAR) LEN(4)
             DCL        VAR(&NUM) TYPE(*CHAR) LEN(4)
             DCL        VAR(&NUMLEN) TYPE(*CHAR) LEN(4)
```

In the example program, two fields are passed to the CL program

The first field is called TEXT that has a fixed length of 15. Data can be moved to it and from it using the 5th argument &TEXT. If it is not a fixed-length CHAR field, then the data length must be used to determine the actual length i.e. field &TEXTLEN.

Unfortunately, the CL language does not support integers directly. To get the numeric value, the contents of &TEXTLEN must be moved to a packed decimal using a statement like:

```
CHGVAR      VAR(&DECTEXTLEN) VALUE(%BIN(&TEXTLEN 1 4))
```

If the length is changed by the program then the new length has to be moved into the TEXTLEN field using a statement like:

```
CHGVAR      VAR(%BINARY(&TEXTLEN)) VALUE(&DECTEXTLEN)
```

External PL/I Routines

PowerExchange provides sample PL/I code for a CALLPROG program in the UCPEP member of the SRCLIB library.

Platform Support

Use linkage type PLI in the third argument to CALLPROG expression to link to a PL/I language program. PL/I language programs can be used on MVS only. Use a C interface layer to call PL/I programs on Windows and UNIX. Use linkage type OS400 to call PL/I programs on i5/OS.

PL/I programs should be compiled using the Language Environment (LE) PL/I run-time routines. Results are unpredictable for PL/I programs compiled with non-LE PL/I run-time routines.

PL/I Linkage

The first four arguments are always the same and are mandatory. The definitions for the fields vary according to the number of fields and their types.

Example Linkage

```
PROC(NUMBER_ARGUMENTS,
      FAILURE_CODE,
      MESSAGE_BUFFER_PTR,
      MESSAGE_BUFFER_LENGTH,
      TEXT_AREA_PTR,
      TEXT_AREA_LENGTH,
      NUMBER1,
      NUMBER1_LENGTH)
      OPTIONS(FETCHABLE)
      REORDER;
 /*---------------------*/
 /* LINKAGE FROM CALLER */
 /*---------------------*/
 DCL  NUMBER_ARGUMENTS        FIXED BIN(31);
 DCL  FAILURE_CODE            FIXED BIN(31);

 DCL  MESSAGE_BUFFER_PTR      PTR;
 DCL  1 MESSAGE_BUFFER_STR    BASED(ADDR(MESSAGE_BUFFER_PTR)),
        4 MESSAGE_BUFFER      CHAR(255);
 DCL  MESSAGE_BUFFER_LENGTH   FIXED BIN(31);

 DCL  TEXT_AREA_PTR           PTR;
 DCL  1 TEXT_AREA_STR         BASED(ADDR(TEXT_AREA_PTR)),
        4 TEXT_AREA           CHAR(15);
 DCL  TEXT_AREA_LENGTH        FIXED BIN(31);

 DCL  NUMBER1                 FIXED BIN(31);
 DCL  NUMBER1_LENGTH          FIXED BIN(31);
```

In the example program, two fields are passed to the program.

- The first field is called TEXT and has a maximum size of 15 bytes. Data can be moved from and to it using PL/1 field TEXT-AREA. If it is not a fixed-length field, then the TEXT-AREA-LENGTH must be used to determine the actual length. If the program wants the length to change, it must store the required value in field TEXT-AREA-LENGTH.
- The second field is numeric, which in the Navigator is defined as NUM32. Data can be moved from and to it using field NUMBER1. It would be unusual to want to make use of field NUMBER1-LENGTH.

  If the field is not null-able, then the field would contain the value 4 on entry to the PL/1 program, which should leave the value untouched.

  NUMBER1-LENGTH is set to zero if the field is NULL. This might happen if the field was NULL before the program was called. If the program decides to make the field NULL, it move zero to NUMBER1-LENGTH.

Using SQL with User-Defined Fields

You should not use the SQL keyword DISTINCT if the data map includes user-defined fields. Attempting to use DISTINCT with user-defined fields causes the following message in the log file:

```
SELECT DISTINCT not guaranteed with Expressions.
```

If DISTINCT is essential to a query then the best approach is to create another table in the data map that contains only the fields that are actually required for that query.

For more information about using SQL in data maps, see the *PowerExchange Reference Manual*.

CHAPTER 11

Creating User Access Method Programs

This chapter includes the following topics:

User Access Method Program Overview

The custom user access method program has a single entry point and is passed a parameter list with every call. The parameter list contains all the parameters specified in the PowerExchange Navigator's data map. The Access module is called repeatedly to process the file. For more information, see “Explanation of Parameter Structure” on page 180.

To show how simple a custom access program can be, the following notes are meant to show in high-level or pseudo-code how the program is written to call a user program called DECOMP, which just happens to read records and decompress the data.

PowerExchange makes the following calls to the custom access program:

Call Type	Custom Access Program Action
OPEN	Get Storage for row to be returned in: - Connect to and open data file - Return with success or fail to PowerExchange
READ	Start or continue sequential reading: - Read the data file to retrieve next record - Make data available to PowerExchange Navigator - Return to PowerExchange with return code (0, 255 or other)
CLOSE	Release acquired storage: - Close data file - Return to PowerExchange

Explanation of Input and Return Parameter

PowerExchange Navigator provides a single input parameter. The user program returns a function return code.

User Program Input Parameter

PowerExchange Navigator provides a single parameter that locates the parameter structure. For more information, see "Explanation of Parameter Structure" on page 180.

Return Codes

The user program returns a function return code. The PowerExchange Navigator expects one of the following return codes:

Return Code	Meaning
0	Mode completed successfully, such as OPENed correctly.
255	No more records available to read, such as end of file.
other	Mode failed.

Explanation of Parameter Structure

The following table describes the parameter structure:

Variable name	Description	PowerExchange Navigator User Interface	Type/Size
LGMODE	Used in connection with LGCODE VALUE 1. and indicates how the data file should be opened. - 1. Read. - 2. Write. - 3. Update. - 4. Append.	Not applicable	Binary/Word
LGCODE	The function to be performed by the USER program. - 1. Open. - 2. Close. - 3. Read. - 4. Write.	Not applicable	Binary/Word
LGUI1	User-defined parameter, for example, record length or number of rows to be retrieved.	Integer 1 (USER Access Method dialog box)	Binary/Word
LGUI2	User-defined parameter, for example, record length or number of rows to be retrieved.	Integer 2 (USER Access Method dialog box)	Binary/Word
LGREADCT	Number of records read from the data file.	Not applicable	Binary/Word
LGWRITCT	Number of records written to the data file.	Not applicable	Binary/Word
LGMLRECL	Record length of the retrieved data.	Set by User program	Binary/Word

Variable name	Description	PowerExchange Navigator User Interface	Type/Size
LGU1LEN	Length of the data entered in String 1	String 1 (USER Access Method dialog box)	Binary/Word
LGU2LEN	Length of the data entered in String 2	String 2 (USER Access Method dialog box)	Binary/Word
LGFNLEN	Length of the data file name	File Name (USER Access Method dialog box)	Binary/Word
LGUIDLEN	Length of the User ID	User ID (on Data Map Remote Node dialog box)	Binary/Word
LGPWDLEN	Length of the Password	Password (on Data Map Remote Node dialog box)	Binary/Word
LGSQLLEN	Length of SQL string	SQL Statement (on Database Row Test dialog box)	Binary/Word
LGU1PTR	Pointer to the data stored for String 1	String 1	Pointer/Word
LGU2PTR	Pointer to the data stored for String 2	String 2	Pointer/Word
LGFNPTR	Pointer to the data stored for the File Name	File Name	Pointer/Word
LGUIDPTR	Pointer to the data stored for the User ID	User ID (on Data Map Remote Node dialog box)	Pointer/Word
LGPWDPTR	Pointer to the data stored for the Password	Password (on Data Map Remote Node dialog box)	Pointer/Word
LGSQLPTR	Pointer to the data stored for SQL string	SQL Statement (on Database Row Test dialog box)	Pointer/Word
LGPUWK1	Pointer to area of storage for user-defined data	Set by User program	Binary/Word
LGPUWK2	Pointer to area of storage for user-defined data	Set by User program	Binary/Word
LGOPENRC	Return code of OPEN function - 1. Successful. - 0. Unsuccessful.	Set by User program	Binary/Word
LGRECLEN	Length of the retrieved record	Set by User program	Binary/Word
LGRECPTR	Pointer to the retrieved record data	Set by User program	Pointer/Word
LGMSGBUF	Communications message returned to PowerExchange from the User program.	Set by User program	Character/80 bytes

Example Programs

Example programs are supplied for Assembler, C, COBOL and PL/1. Each example demonstrates the following logic.

1. The program receives an OPEN call. For the OPEN call, the custom access module might want to perform some or all of the following actions:
   - Obtain working storage.
   - Validate and save any passed parameters.
   - Initialize counters.
   - Check the file exists for reading and open it.
   - Erase the output file if the target is to be replaced.
2. The program receives a READ or WRITE call.
   - **READ call**. On a read call, the access module passes data back to the calling routine by setting the record pointer to point to the data and set the record_len field to the length of the data being returned. When the last record has been processed, the custom access module returns a return code of 255 to signal end of file. Any return code other than 0 and 255 is treated as an error. It is the responsibility of the access module to read the file or database and build the input record. The custom access module might need to examine the SQL statement passed from PowerExchange or the ETL tool to create the desired record.
   - **WRITE call**. When writing to a file, the data is passed by the record pointer and the length is set in the record_len field. The access module can do whatever it deems necessary with the data. A 0 return code signals a successful write.
3. Finally, the access module receives a CLOSE call. During CLOSE processing, the program might perform some or all of these tasks:
   - Close the file.
   - Release storage.
   - Commit the database.
   - Set a return code. If all tasks complete successfully, the module returns a return code of 0. However, if an error occurs, the module sets a non-zero return code, and places message text in the message buffer. The message buffer holds a 79 character message.
   - Pass control back to the calling program.
4. The access module receives is the user ID and password from the PowerExchange Navigator screen or the ETL tool. The access module uses these credentials to check file security.

Assembler Example

```
*********************************************************************   00010000
*                                                                     00020000
* USREX002 - MVS Assembler Interface                                  00030000
* This is a sample exit that demonstrates some of the usage of        00031016
* exit USREX002. It mainly produces diagnostic messages that          00032016
* show calling and returned values rather than reading and writing    00033016
* real user data.                                                     00034016
*                                                                     00035016
* N.B. OS calling linkage is used with return code always in R15      00040000
*      and that the calling program is written in C hence the use     00050000
*      of EDCPRLG and EDCEPIL at the beginning and end of the         00060000
*      exit. These macros should be in CEE.SCEEMAC or your local      00070001
*      equivalent.                                                    00080001
*                                                                     00090000
* 1 Entry point as follows:                                           00100000
*                                                                     00110000
* USREX002                                                            00120000
* --------                                                            00130000
* int USREX002(char *lgif);                                           00140000
* User exit to handle I/O related requests                            00150000
*                                                                     00160000
* Return Codes in R15:                                                00170000
*      255        End of file                                         00180000
```

```
*        0        OK                                                    00200000
*      other      non specific error - NULL delimited message in        00201016
*                                      LGMSGBUF (Max 79 chars + NULL)   00202016
*                                                                       00203016
*                                                                       00210000
* The details of the request are passed to and from the exit via        00220000
* a structure whose address is passed as the single parameter to        00230000
* this exit. The  DSECT for this is provided separately and may         00240014
* be copied in thus:                                                    00250014
         COPY  USREX02D                                                 00260014
*                                                                       00270000
* This test exit assumes that a SYSDIAGS dataset is allocated and uses  00280023
* this dataset with an LRECL = 120.                                     00290023
* So in your JCL you will need something like                           00310023
* //SYSDIAGS DD SYSOUT=A                                               00320023
*                                                                       00330016
*                                                                       00340016
*                                                                       00350016
*                                                                       00360016
*                                                                       00370016
************************************************************************ 00380000
*                                                                       00381016
* Two macros are defined inline to invoke the C Prolog and Epilog       00382016
* functions.                                                            00383016
*                                                                       00384016
         MACRO                                                          00390000
&NAME    FUNCSTR                                                        00400000
&NAME    AMODE ANY                                                      00410000
&NAME    RMODE ANY                                                      00420000
&NAME    CSECT                                                          00430000
         EDCPRLG  BASEREG=12                                            00440000
         MEND                                                           00450000
*                                                                       00460000
         MACRO                                                          00470000
&NAME    FUNCEND                                                        00480000
         EDCEPIL                                                        00490000
         MEND                                                           00500000
*                                                                       00510000
****** USREX002 Entry Point                                            00520016
*                                                                       00530000
USREX002 FUNCSTR                                                        00540000
         L     R3,0(R1)               get structure address           00550001
         USING LGIF,R3                                                  00560001
         GETMAIN R,LV=WORKLEN         get some work storage             00570016
         LR    R4,R1                  save the address                  00580001
         USING WORKAREA,R4                                              00590001
         XR    R15,R15                clear RC                          00600002
         ST    R15,WORKRC             set a good one for now            00610002
*                                                                       00620002
*** End of prologue ***                                                 00630002
*                                                                       00640002
*                                                                       00641016
*** set up to use SYSDIAGS dataset for test messages                    00642024
*                                                                       00643016
         MVC   WORKOPEN(LOPENLEN),LOPEN copy OPEN prototype             00650016
         MVC   WORKCLOS(LCLOSLEN),LCLOS copy CLOSE prototype            00660016
         MVC   WORKDCB(LDCBLEN),LDCB    copy DCB prototype              00670016
         OPEN  (WORKDCB,(OUTPUT)),MF=(E,WORKOPEN)                       00680006
         PUT   WORKDCB,=CL120'00001 ENTERING EXIT USREX002'             00690009
*                                                                       00700009
*** Format the input structure ***                                      00710016
*                                                                       00720009
         PUT   WORKDCB,=CL120'00002 INPUT VALUES IN HEX'                00730009
         MVC   WORKLINE(LGIFLEN),LGIF      translate                    00740016
         TR    WORKLINE(LGIFLEN),DUMPTBHI  the high hex digits          00750016
         PUT   WORKDCB,WORKLINE            and print them               00760016
         MVC   WORKLINE(LGIFLEN),LGIF      translate the                00770016
         TR    WORKLINE(LGIFLEN),DUMPTBLO  low hex digits               00780016
         PUT   WORKDCB,WORKLINE            and print them               00790016
*                                                                       00800013
*** FORMAT SOME FIELDS OF INTEREST ****                                 00810013
*                                                                       00820013
         L     R5,LGFNLEN             get the length of the file name  00830016
         LTR   R5,R5                  is there a file name ?           00840014
         BZ    NOFILENM               no                               00850016
         C     R5,=F'120'             is it too long ?                 00860014
         BNH   PRFILENM               no                               00870014
         L     R5,=F'120'             Yes do the first 120             00880014
PRFILENM DS    0H                                                      00890014
         PUT   WORKDCB,=CL120'00005 THE FILE NAME SPECIFIED IS'         00900015
         MVI   WORKLINE,C' '          clear the print                  00910016
         MVC   WORKLINE+1(119),WORKLINE   buffer                       00920016
         L     R6,LGFNPTR             get the pointer                  00930014
         LTR   R6,R6                  is it OK ?                        00930116
         BZ    NOFILENM               Dont bother.                     00930216
         BCTR  R5,0                   for the executed move            00931020
```

```
         EX    R5,MVCFILE              move the filename               00940014
         PUT   WORKDCB,WORKLINE        print it                        00950014
         B     NOFILENM                                                00960014
MVCFILE  MVC   WORKLINE(*-*),0(R6)                                     00970014
NOFILENM DS    0H                                                      00980014
*                                                                      00990010
*** Test the input opcode to see what function is requested            01000016
*                                                                      01010010
         CLC   LGCODE(4),=AL4(LGOPEN)   is this an OPEN ?              01020007
         BE    DOOPEN                                                  01030003
         CLC   LGCODE(4),=AL4(LGCLOSE)  is this a CLOSE ?              01040007
         BE    DOCLOSE                                                 01050003
         CLC   LGCODE(4),=AL4(LGREAD)   is this a READ ?               01060007
         BE    DOREAD                                                  01070003
         CLC   LGCODE(4),=AL4(LGWRITE)  is this a WRITE ?              01080007
         BE    DOWRITE                                                 01090003
         MVC   WORKRC,=F'4'             whats going on ?               01100003
         XC    LGMSGBUF,LGMSGBUF                                       01101016
         MVC   LGMSGBUF(40),=CL40'Invalid OPCODE specified'            01102016
         B     EXIT                                                    01110003
*                                                                      01120003
*** DOOPEN ***                                                         01130003
*                                                                      01140003
DOOPEN   DS    0H                                                      01150003
         CLC   LGMODE(4),=AL4(LGRMODE)  is this an OPEN for read ?     01160007
         BE    OPENREAD                                                01170004
         CLC   LGMODE(4),=AL4(LGWMODE)  is this an OPEN for write ?    01180007
         BE    OPENWRIT                                                01190004
         CLC   LGMODE(4),=AL4(LGUMODE)  is this an OPEN for update ?   01200007
         BE    OPENUPDT                                                01210004
         CLC   LGMODE(4),=AL4(LGAMODE)  is this an OPEN for append ?   01220007
         BE    OPENAPPN                                                01230004
         MVC   WORKRC,=F'4'             whats going on ?               01240004
         XC    LGMSGBUF,LGMSGBUF                                       01241016
         MVC   LGMSGBUF(40),=CL40'Invalid OPEN mode specified'         01242016
         B     EXIT                                                    01250004
*                                                                      01260004
*** For testing we merely set the file open flag ***                   01270016
*                                                                      01280004
OPENREAD DS    0H                                                      01290004
         PUT   WORKDCB,=CL120'00011 OPCODE=1 MODE=1 Opening for Read'  01300011
         B     OPENEND                                                 01310011
OPENWRIT DS    0H                                                      01320004
         PUT   WORKDCB,=CL120'00012 OPCODE=1 MODE=2 Opening for Write' 01330011
         B     OPENEND                                                 01340011
OPENUPDT DS    0H                                                      01350004
         PUT   WORKDCB,=CL120'00013 OPCODE=1 MODE=3 Opening for Update'01360011
         B     OPENEND                                                 01370011
OPENAPPN DS    0H                                                      01380004
         PUT   WORKDCB,=CL120'00014 OPCODE=1 MODE=4 Opening for Append'01390011
         B     OPENEND                                                 01400011
OPENEND  DS    0H                                                      01410011
         MVC   LGOPENRC,=CL4'OPEN'     set file open                   01420010
         B     EXIT                                                    01430004
*                                                                      01440004
*** DOCLOSE ***                                                        01450004
*                                                                      01460004
* for testing CLOSE we merely reset the file open flag                 01461016
*                                                                      01462016
DOCLOSE  DS    0H                                                      01470004
         PUT   WORKDCB,=CL120'00020 OPCODE=2 Closing the file'         01480011
         XC    LGOPENRC,LGOPENRC       set file close                  01490010
         B     EXIT                                                    01500010
*                                                                      01510004
*** DOREAD ***                                                         01520004
*                                                                      01530004
* for read testing return one of our hard coded records               01531016
* index down the array on the basis of the read count                 01532016
* specified in the input structure. 0 returns the first record        01533016
*                                                                      01534016
DOREAD   DS    0H                                                      01540004
         PUT   WORKDCB,=CL120'00030 OPCODE=3 Doing a Read'             01550011
         L     R5,LGREADCT             Read count from input           01560016
         C     R5,NUMRECS              too many ?                      01561016
         BH    NOMORE                  YES                             01562016
         MH    R5,=H'80'               index down our samples          01570016
         LA    R6,FIRST(R5)            no                              01610016
         ST    R6,LGRECPTR             return the pointer              01620016
         MVC   LGRECLEN,LENGTH         and length                      01630016
         B     EXIT                                                    01640011
NOMORE   DS    0H                                                      01650015
         PUT   WORKDCB,=CL120'00031 End of file on READ'               01651022
         XC    LGRECLEN,LGRECLEN       EOF so clear the length         01660016
         XC    LGRECPTR,LGRECPTR       clear the pointer               01670016
         MVC   WORKRC,=F'255'          set the flag                    01680016
         B     EXIT                                                    01690015
```

```
*                                                                       01700010
*** DOWRITE ***                                                         01710004
*                                                                       01720004
* for testing just output the first 120 bytes of the record            01730004
*                                                                       01731016
DOWRITE  DS    0H                                                       01732016
         PUT   WORKDCB,=CL120'00040 OPCODE=4 Doing a Write of...'       01740004
         L     R5,LGRECLEN            Get the length to be written     01750016
         L     R6,LGRECPTR            Get address of data              01751016
         LTR   R5,R5                  anything to write ?              01752016
         BZ    EXIT                                                    01753016
         LTR   R6,R6                  any buffer ?                     01754016
         BZ    EXIT                                                    01755016
         MVI   WORKLINE,C' '          clear the print line             01756016
         MVC   WORKLINE+1(119),WORKLINE                                01757016
         C     R5,=F'120'             too much to print ?              01758016
         BNH   WRTPRINT               no                               01758116
         L     R5,=F'120'             yes set the maximum              01759016
WRTPRINT DS    0H                                                       01759116
         BCTR  R5,0                   set up the move                  01759216
         EX    R5,MVCFILE             do it and                        01759320
         PUT   WORKDCB,WORKLINE       print                            01759416
*                                                                       01759516
*** EXIT ***                                                           01760002
*                                                                       01770002
EXIT     DS    0H                                                       01780002
*                                                                       01790001
*** test returned message facility ***                                 01791016
*                                                                       01792016
* Just for test purposes if LGUI1 is set non-zero                      01793016
* we return a null delimited msg.                                      01793116
*                                                                       01793216
         L     R5,LGUI1            shall we return a message string ? 01793316
         LTR   R5,R5                                                   01794016
         BZ    NOMSG               No msg to return                    01795016
         XC    LGMSGBUF,LGMSGBUF                                       01796016
         MVC   LGMSGBUF(40),=CL40'Just a test message'                 01796116
         MVC   WORKRC,=F'4'       tell main line there's a msg         01796216
NOMSG    DS    0H                                                       01796316
*                                                                       01797016
***                                                                    01800010
*                                                                       01810016
         PUT   WORKDCB,=CL120'00003 LEAVING EXIT USREX002'             01820010
*                                                                       01830010
*** format the returned structure in hex ***                           01840010
*                                                                       01850016
         PUT   WORKDCB,=CL120'00004 OUTPUT VALUES IN HEX'              01860010
         MVC   WORKLINE(LGIFLEN),LGIF      format the                  01870010
         TR    WORKLINE(LGIFLEN),DUMPTBHI  high hex digits             01880016
         PUT   WORKDCB,WORKLINE            and print                   01890016
         MVC   WORKLINE(LGIFLEN),LGIF      format the low              01900016
         TR    WORKLINE(LGIFLEN),DUMPTBLO  hex digits                  01910016
         PUT   WORKDCB,WORKLINE            and print                   01920016
         CLOSE WORKDCB,MF=(E,WORKCLOS)     close the diagnostic file   01930016
*                                                                       01940016
***                                                                    01950010
*                                                                       01960010
         L     R5,WORKRC                   pick up the RC value        01970010
         DROP  R4                                                      01980016
         FREEMAIN R,LV=WORKLEN,A=(4)   free work area                  01990001
         LR    R15,R5                      put RC in the correct reg.  02000001
         FUNCEND                                                       02010016
*                                                                       02020000
*** some sample records to return                                      02030004
*                                                                       02040004
SAMPRECS DS    0F                                                       02050004
LENGTH   DC    F'80'                                                    02060010
NUMRECS  DC    F'2'         actually 3 but input record count starts    02070010
*                           at 0                                       02071016
FIRST    DC    CL80'WIBBLE1'                                            02072016
         DC    CL80'WIBBLE2'                                            02080010
         DC    CL80'WIBBLE3'                                            02090010
EOF      DC    F'00'                                                    02100010
*                                                                       02110010
*** constants ***                                                      02120004
*                                                                       02130004
LOPEN    OPEN  (,(OUTPUT)),MF=L                                        02140004
LOPENLEN EQU   *-LOPEN                                                 02150005
LCLOS    CLOSE (,),MF=L                                                02160004
LCLOSLEN EQU   *-LCLOS                                                 02170007
LDCB     DCB   DDNAME=SYSDIAGS,DSORG=PS,MACRF=PM,LRECL=120             02180007
LDCBLEN  EQU   *-LDCB                                                  02190024
DUMPTBHI DC    16X'F0'                                                 02200004
         DC    16X'F1'                                                 02210009
         DC    16X'F2'                                                 02220009
                                                                       02230009
```

```
         DC    16X'F3'                                                  02240009
         DC    16X'F4'                                                  02250009
         DC    16X'F5'                                                  02260009
         DC    16X'F6'                                                  02270009
         DC    16X'F7'                                                  02280009
         DC    16X'F8'                                                  02290009
         DC    16X'F9'                                                  02300009
         DC    16X'C1'                                                  02310009
         DC    16X'C2'                                                  02320009
         DC    16X'C3'                                                  02330009
         DC    16X'C4'                                                  02340009
         DC    16X'C5'                                                  02350009
         DC    16X'C6'                                                  02360009
DUMPTBLO DC    16X'F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'                    02370009
         LTORG                                                          02380000
WORKAREA DSECT                                                          02390001
WORKRC   DS    F                                                        02400002
WORKLINE DS    CL133                                                    02410009
WORKLINF DS    CL133                                                    02420009
WORKOPEN DS    CL(LOPENLEN)                                             02430004
WORKCLOS DS    CL(LCLOSLEN)                                             02440007
WORKDCB  DS    CL(LDCBLEN)                                              02450004
WORKLEN  EQU   *-WORKAREA                                               02460001
*                                                                      02470000
R0       EQU   0                                                       02480000
R1       EQU   1                                                       02490000
R2       EQU   2                                                       02500000
R3       EQU   3                                                       02510000
R4       EQU   4                                                       02520000
R5       EQU   5                                                       02530000
R6       EQU   6                                                       02540000
R7       EQU   7                                                       02550000
R8       EQU   8                                                       02560000
R9       EQU   9                                                       02570000
R10      EQU   10                                                      02580000
R11      EQU   11                                                      02590000
R12      EQU   12                                                      02600000
R13      EQU   13                                                      02610000
R14      EQU   14                                                      02620000
R15      EQU   15                                                      02630000
         END                                                           02640000
```

C Example

```
/*************************************************************************/
/* USREX001.C                                                            */
/* Contains an example of a user exit program                            */
/*************************************************************************/
#include "stdio.h"
#include "string.h"

#include "dtlam1gi.h"

#if defined WIN32
   #define DLLEXPORT    __declspec(dllexport)
#else
   #define DLLEXPORT
#endif

#if defined DTL_MVS
    #pragma linkage(USREX001, fetchable)
#endif

/*************************************************************************/
/* Internal prototypes                                                   */
/*************************************************************************/
static int F01_Open(pLGIF plgif);
static int F02_Close(pLGIF plgif);
static int F03_Read(pLGIF plgif);
static int F04_Write(pLGIF plgif);

DLLEXPORT int Print_Parms(pLGIF plgif);

/*************************************************************************/
/* Dummy file                                                           */
/*************************************************************************/

#define MAXIND 6

char * dummy_file[MAXIND]=
{
 "Barcelona",
 "Manchester United",
 "Bayer",
```

```
 "Juventus",
 "Arsenal",
 "Inter"
};

/*=========================================================================*/
/* USREX001                                                                */
/* This is the only external entry point to this module                    */
/*=========================================================================*/
#if defined DTL_MVS
DLLEXPORT int USREX001(void *pinterface)
#else
DLLEXPORT int usrex001(void *pinterface)
#endif
{
   pLGIF plgif = (pLGIF) pinterface;

   switch (plgif->opcode)
          {
           case LGIF_OPEN:
                return F01_Open(plgif);
   case LGIF_CLOSE:
                return F02_Close(plgif);
   case LGIF_READ:
                return F03_Read(plgif);
            case LGIF_WRITE:
                return F04_Write(plgif);
            default:
                printf("Invalid Opcode\n");
          }
   return 0;
}
/*=========================================================================*/
/* F01_Open                                                                */
/* Open file for input.                                                    */
/*=========================================================================*/
static int F01_Open(pLGIF plgif)
{
   printf("** OPEN being called\n");
   Print_Parms(plgif);
   /* Demonstrate returning an error message if ui1 = 99 */
   if (plgif->ui1 == 99)
      {
       strcpy(plgif->umsgbuf, "Attempt to start USREX001 with invalid ui1");
       return 77;  /* User specified return code */
      }
   plgif->file_opened = 1;   /* set to !=0 if opened OK     */
   plgif->puwk1 = (char) 0;  /* initialize the file pointer */

   return 0;
}
/*=========================================================================*/
/* F02_Close                                                               */
/*=========================================================================*/
static int F02_Close(pLGIF plgif)
{
   printf("** CLOSE being called\n");
   Print_Parms(plgif);

   return 0;
}
/*=========================================================================*/
/* F03_Read                                                                */
/*=========================================================================*/
static int F03_Read(pLGIF plgif)
{
   int i = (int) (plgif->puwk1);
   printf("** READ being called\n");
   Print_Parms(plgif);
   if (i >= MAXIND)        /* if end of file, return 255 */
        return (255);
   else
   {
 plgif->prec=dummy_file[i];
 plgif->reclen = strlen(dummy_file[i]);
 i ++;
 plgif->puwk1 = (char *) i;
   }

   return 0;
}
/*=========================================================================*/
/* F04_Write                                                               */
/* Just write the output record.                                           */
/*=========================================================================*/
static int F04_Write(pLGIF plgif)
```

```
{
   printf("** WRITE being called\n");
   Print_Parms(plgif);

   return 0;
}

/*====================================================================*/
/* Print                                                              */
/*====================================================================*/
DLLEXPORT int Print_Parms(pLGIF plgif)
{
/* Print opcode */
switch (plgif->opcode)
          {
           case LGIF_OPEN:
                printf("        opcode  = %s\n","OPEN");
break;
   case LGIF_CLOSE:
                printf("        opcode  = %s\n","CLOSE");
                break;
   case LGIF_READ:
                printf("        opcode  = %s\n","READ");
                break;
           case LGIF_WRITE:
                printf("        opcode  = %s\n","WRITE");
                break;
           default:
                printf("        opcode  = %d\n",plgif->opcode);
          }
    /* Print openmode */
switch (plgif->openmode)
          {
           case LGIF_WRITEMODE:
                printf("        openmode= %s\n","WRITEMODE");
                break;
           case LGIF_UPDATEMODE:
                printf("        openmode= %s\n","UPDATEMODE");
                break;
           case LGIF_APPENDMODE:
                printf("        openmode= %s\n","APPENDMODE");
                break;
   default:
                printf("        openmode= %d\n",plgif->openmode);
                break;
   }
   if (plgif->opcode == LGIF_OPEN)
   {
     printf("        ui1     = %d\n",plgif->ui1);
     printf("        ui2     = %d\n",plgif->ui2);
   }
   if (plgif->puser1 != NULL)
         printf("        user1   = %s\n",plgif->puser1);
   if (plgif->puser2 != NULL)
         printf("        user2   = %s\n",plgif->puser2);
   if (plgif->pfn != NULL)
         printf("        pfn     = %s\n",plgif->pfn);
   printf("        puwk1   = %d\n",(int) plgif->puwk1);
   printf("        reclen  = %d\n",plgif->reclen);
   printf("        rec     = %s\n",plgif->prec);
   printf(" \n");

   return 0;
}
```

COBOL Example

```
//UAMCOB JOB (),CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),REGION=4M,
//        NOTIFY=&SYSUID
//*
//          SET HLQ=<libname>
//          SET VER=<version>
//*************************************************************
//* COMPILE PROGRAM
//COBCOMP EXEC PROC=IGYWCL,PARM=('XREF(FULL)')
//COBOL.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. UAMCOB.
      *REMARKS. THIS IS A SAMPLE PRGRAM WHICH EXPECTS TO BE DRIVEN
      *         BY A DATAMAP WHICH INVOKES PROGRAM UAMCOB.
      *         THE MAP MUST HAVE IN INTEGER 1 THE LENGTH OF THE
      *         MAXIMUM RECORD AND IN INTEGER 2 THE NUMBER OF RECS
      *         TO BE READ.
```

```
 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77  DEBUG-SW                        PIC X VALUE '0'.
 77  HEAP0                           PIC S9(8) COMP VALUE +0.
 77  RECCT                           PIC S9(8) COMP VALUE +0.
 01  STOR-C.
     04   STOR-C-RC                  PIC X(4).
     04   STOR-C-8                   PIC X(8).

 LINKAGE SECTION.
 01  LGIF.
* For the OPEN request, the OPEN mode type requested may be one of
* the following
     04  LGMODE                  PIC S9(8) COMP.
         88 LGRMODE VALUE 1.
         88 LGWMODE VALUE 2.
         88 LGUMODE VALUE 3.
         88 LGAMODE VALUE 4.
* User exit may handle four types I/O related requests:
     04  LGCODE                  PIC S9(8) COMP.
         88 LGOPEN  VALUE 1.
         88 LGCLOSE VALUE 2.
         88 LGREAD  VALUE 3.
         88 LGWRITE VALUE 4.
     04  LGUI1                   PIC S9(8) COMP.
     04  LGUI2                   PIC S9(8) COMP.
     04  LGREADCT                PIC S9(8) COMP.
     04  LGWRITCT                PIC S9(8) COMP.
     04  LGMLRECL                PIC S9(8) COMP.
     04  LGU1LEN                 PIC S9(8) COMP.
     04  LGU2LEN                 PIC S9(8) COMP.
     04  LGFNLEN                 PIC S9(8) COMP.
     04  LGUIDLEN                PIC S9(8) COMP.
     04  LGPWDLEN                PIC S9(8) COMP.
     04  LGSQLLEN                PIC S9(8) COMP.
     04  LGU1PTR                 POINTER.
     04  LGU2PTR                 POINTER.
     04  LGSQLPTR                POINTER.
     04  LGFNPTR                 POINTER.
     04  LGUIDPTR                POINTER.
     04  LGPWDPTR                POINTER.
     04  LGPUWK1                 PIC S9(8) COMP.
     04  LGPUWK2                 PIC S9(8) COMP.
     04  LGOPENRC                PIC S9(8) COMP.
     04  LGRECLEN                PIC S9(8) COMP.
     04  LGRECPTR                POINTER.
     04  LGMSGBUF                PIC X(80).

 01  RECORD-OUT.
     04   DESC                       PIC X(9).
     04   RECCT1                     PIC Z(6)9.
     04   RECCT2                     PIC S9(4) COMP.
     04   RECCT3                     PIC S9(8) COMP.
     04   RECCT4                     PIC S9(7) COMP-3.
 01 RC                  PIC S9(9) USAGE BINARY.

 PROCEDURE DIVISION USING LGIF RETURNING RC.

  S0001-MAIN SECTION.
     IF LGOPEN
        PERFORM S1000-OPEN
        GO TO S0001-P99.
     IF LGCLOSE
        PERFORM S3000-CLOSE
        GO TO S0001-P99.
     IF LGREAD
        PERFORM S2000-READ
        GO TO S0001-P99.
     MOVE 4 TO RC.
     MOVE 'Invalid OPCODE specified' TO LGMSGBUF.
  S0001-P99.
     GOBACK.
*****************************************************************
  S1000-OPEN SECTION.
          IF DEBUG-SW = 1 DISPLAY 'UAMCOB IN OPEN'.
          MOVE 1 TO LGOPENRC.
          MOVE 0 TO RC.
          MOVE SPACES TO LGMSGBUF.
          CALL 'CEEGTST' USING HEAP0,
                               LGUI1,
                               LGRECPTR,
                               STOR-C.
          IF STOR-C-RC NOT = LOW-VALUES
              MOVE 4 TO RC
```

```
                ELSE
                 SET ADDRESS OF RECORD-OUT TO LGRECPTR.
           EXIT.
      ***************************************************************
        S2000-READ SECTION.
                IF DEBUG-SW = 1 DISPLAY 'UAMCOB IN READ'.
                IF LGREADCT >= LGUI2
                     MOVE 0 TO LGRECLEN
                     MOVE 255 TO RC
                     DISPLAY 'END OF FILE SET'
                     GO TO S2000-P99.
                MOVE 80 TO LGRECLEN.
                MOVE 0 TO RC.
                MOVE 'RECORD ' TO DESC.
                ADD 1 TO RECCT.
                MOVE  RECCT       TO RECCT1
                                     RECCT2
                                     RECCT3
                                     RECCT4.
       S2000-P99.
             EXIT.
      ***************************************************************
        S3000-CLOSE SECTION.
                IF DEBUG-SW = 1 DISPLAY 'UAMCOB IN CLOSE'.
                IF LGRECPTR NOT = NULL
                   CALL 'CEEFRST' USING LGRECPTR,
                                        STOR-C.
                MOVE 0 TO LGOPENRC.
                MOVE 0 TO RC.
                MOVE SPACES TO LGMSGBUF.
             EXIT.
      ***************************************************************
        END PROGRAM UAMCOB.
/*
// LKED.SYSLMOD  DD  DISP=SHR,DSN=&HLQ..&VER..LOADLIB
//LKED.SYSIN    DD *
 ENTRY UAMCOB
 NAME UAMCOB(R)
/*
```

23.4.4PL/1 example

```
*PROCESS OPTIONS NOINSOURCE SOURCE NEST MACRO MAP STORAGE;
*PROCESS AGGREGATE, ESD, OFFSET;
*PROCESS LIST(1,999) FLAG(I) MARGINS(2,72,1) MARGINI(']');
*PROCESS OPT(0) TEST(ALL,SYM) ATTRIBUTES(FULL) XREF(SHORT);
 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  * PROGRAM  : UAMPL1                                                    *
  * REMARKS  : THIS IS A SIMPLE PROGRAM WHICH EXPECTS TO BE DRIVEN    *
  *            BY A DATAMAP WHICH INVOKES IT. THE DATAMAP MUST HAVE   *
  *            IN INTEGER 1 THE LENGTH OF THE MAXIMUM RECORD (NOT <   *
  *            26) AND IN INTEGER 2 THE NBR OF RECORDS TO BE READ.    *
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

 UAMPL1:
 PROC(LGIF_PTR) OPTIONS(FETCHABLE) RETURNS(FIXED BIN(31));

 /* INPUT PARAMETER AND DEFINITION OF                                   */
 DCL  LGIF_PTR                PTR;
 DCL  1 LGIF                  BASED(ADDR(LGIF_PTR)),
        4 LGMODE              FIXED BIN(31),
        4 LGCODE              FIXED BIN(31),
        4 LGUI1               FIXED BIN(31),
        4 LGUI2               FIXED BIN(31),
        4 LGREADCT            FIXED BIN(31),
        4 LGWRITCT            FIXED BIN(31),
        4 LGMLRECL            FIXED BIN(31),
        4 LGU1LEN             FIXED BIN(31),
        4 LGU2LEN             FIXED BIN(31),
        4 LGFNLEN             FIXED BIN(31),
        4 LGUIDLEN            FIXED BIN(31),
        4 LGPWDLEN            FIXED BIN(31),
        4 LGSQLLEN            FIXED BIN(31),
        4 LGU1PTR             PTR,
        4 LGU2PTR             PTR,
        4 LGSQLPTR            PTR,
        4 LGFNPTR             PTR,
        4 LGUIDPTR            PTR,
        4 LGPWDPTR            PTR,
        4 LGPUWK1             FIXED BIN(31),
        4 LGPUWK2             FIXED BIN(31),
        4 LGOPENRC            FIXED BIN(31),
        4 LGRECLEN            FIXED BIN(31),
        4 LGRECPTR            PTR,
        4 LGMSGBUF            CHAR(80);
```

```
/* GENERAL DECLARATIONS AND BUILTIN FUNCTIONS.                        */
DCL  RC                      FIXED BIN(31) INIT(0);

/* OUTPUT RECORD AREA (FIXED FOR EXAMPLE PURPOSE)                     */
DCL  TEMP_STRING             CHAR(*) STATIC  INIT('');
DCL  1 REC_OVERLAY           BASED(ADDR(TEMP_STRING)),
       4 CHAR_ID             CHAR(7),
       4 PIC_NBR             PIC'(6)Z9',
       4 FB_2_NBR            FIXED BIN(15),
       4 FB_4_NBR            FIXED BIN(31),
       4 PACK_NBR            FIXED(7);
DCL  (ADDR,
      MIN,
      VERIFY)                BUILTIN;


/* MAINLINE PROGRAM - DETERMINE ACTION AND PROCESS ACCORDINGLY        */
SELECT(LGIF.LGCODE);
  WHEN(1) DO;                                        /* OPEN FILE      */
          REC_OVERLAY = '';
          REC_OVERLAY.CHAR_ID = 'RECORD';
          LGIF.LGOPENRC = 1;
          LGIF.LGRECLEN = LGIF.LGUI1;
          LGIF.LGRECPTR = ADDR(TEMP_STRING);
  END;
  WHEN(2) DO;                                        /* CLOSE FILE     */
/* PERFORM ANY CLOSE FUNCTIONALITY       */
  END;
  WHEN(3) DO;                                        /* READ RECORD    */
          REC_OVERLAY.PIC_NBR = REC_OVERLAY.PIC_NBR +1;
          REC_OVERLAY.PACK_NBR = REC_OVERLAY.PACK_NBR +1;
          REC_OVERLAY.FB_2_NBR = REC_OVERLAY.FB_2_NBR +1;
          REC_OVERLAY.FB_4_NBR = REC_OVERLAY.FB_4_NBR +1;
          LGIF.LGRECLEN = LGIF.LGUI1;
          IF LGIF.LGREADCT >= LGIF.LGUI2
          THEN DO;
               RC = 255;
               LGIF.LGRECLEN = 0;
          END;
  END;
  OTHER   ;
END;
RETURN(RC);
END UAMPL1;
```

INDEX

D

E

F

T

U

V

W

NOTICES

This Informatica product (the “Software”) includes certain drivers (the “DataDirect Drivers”) from DataDirect Technologies, an operating company of Progress Software Corporation (“DataDirect”) which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.

2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.